
# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

## FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

## ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION


# FPGA MODUL PRO ŘÍZENÍ BLDC MOTORŮ
FPGA BASED CONTROLLER DRIVE OF BLDC MOTOR

**BAKALÁŘSKÁ PRÁCE**
BACHELOR'S THESIS


**AUTOR PRÁCE** — David Makówka
AUTHOR

**VEDOUCÍ PRÁCE** — Ing. Soběslav Valach
SUPERVISOR



**BRNO 2020**

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

# Bakalářská práce

bakalářský studijní program **Automatizační a měřicí technika**

Ústav automatizace a měřicí techniky

***Student:*** David Makówka ***ID:*** 203580
***Ročník:*** 3 ***Akademický rok:*** 2019/20

**NÁZEV TÉMATU:**

## FPGA modul pro řízení BLDC motorů

**POKYNY PRO VYPRACOVÁNÍ:**

Cílem práce je návrh více kanálového řídicího modulu BLDC motorů využívající pro řízení FPGA.
1) Seznámit se s možnostmi řízení BLDC motorů a jejich parametry
2) Navrhnout a ověřit simulací vhodný model řízení
3) Navrhnout schéma zapojení řídicího modulu
4) Připravit návrh desky plošného spoje řídicího modulu
5) Koncepce řízení BLDC motoru v FPGA
6) Návrh metod verifikace funkce a zvolené koncepce

**DOPORUČENÁ LITERATURA:**

[1] Fischer P. High Performance Brushless DC Motor Control. School of Engineering &Technology CQUniversity Australia. May 2014

***Termín zadání:*** 3.2.2020 ***Termín odevzdání:*** 8.6.2020

***Vedoucí práce:*** Ing. Soběslav Valach

**doc. Ing. Václav Jirsík, CSc.**
předseda rady studijního programu

# Abstrakt

Práce se zabývá možnostmi řízení BLDC motorů za využití FPGA čipů a také návrhem měniče pro kompletní realizaci. Pro řízení BLDC motoru byla implementována šestikroková komutace, pro budoucí usnadnění implementace vektorového řízení. Vektorové řízení bylo navrženo v prostředí MATLAB Simulink v semestrální práci. Cílovou platformou byla zvolena FPGA deska Basys 3. Měnič byl navržen pro BLDC motory do 6 A. Ošetření chybových stavů je řešeno především integrovaným obvodem DRV8305. Chyby jsou dále posílány do řídící desky Basys 3, která umožnuje další zásahy do řídícího hardwarového popisu. Struktura řídícího programu je navržena pro snadné ladění parametrů. Umožnuje změnu jednotlivých parametrů za chodu a fáze řízení jsou procházeny jednotlivě. Snímání napětí a proudů fázemi je prováděno za použití analogově-digitálního převodníku.

# Klíčová slova

Řízení, BLDC motor, FPGA, FOC, SVM, PWM, bezsenzorové, trapézové, Parkova, Clarkova, transformace, měnič, převodník, Simulink, MATLAB, regulace, SPI, šestikroková komutace, plošný spoj, Vivado, VHDL, simulace, pájení

# Abstract

This bachelor’s thesis concerns different control approaches for driving a BLDC motor using an FPGA chip. Also, a custom type of an inverter circuit was designed. A six-step commutation control scheme has been implemented, to ease the future integration of field-oriented control. The field-oriented control has been designed and simulated in a semestral thesis using a MATLAB Simulink tool. The targeted platform is the FPGA development board Basys 3. Hardware is rated to deliver up to 6 A of current. The handling of error conditions is mainly provided by a DRV8305 gate driver integrated circuit. Errors are also forwarded to the FPGA, for performing further actions. The structure of a controlling scheme is accustomed to the tuning of motor parameters rather than for end-users. Parameters can be set during motor operation and states of the control scheme are stepped separately. The sensing of voltages and currents is handled by an analog-digital converter.

# Keyword

Control, BLDC motor, FPGA, FOC, SVM, PWM, Sensor less, Trapezoidal, Back EMF, park, Clark, transformation, converter, Vivado, Simulink, MATLAB, regulation, SPI, six step commutation, printed circuit board, Vivado, VHDL, simulation, soldering

# Bibliografická citace

MAKÓWKA, David. FPGA modul pro řízení BLDC motorů [online]. Brno, 2020 [cit. 2020-06-07]. Dostupné z: **https://www.vutbr.cz/studenti/zav-prace/detail/126964**. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Soběslav Valach.

# Prohlášení

Prohlašuji, že svou bakalářskou práci na téma „FPGA modul pro řízení BLDC motorů“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne 8. června 2020 ……..

podpis autora

# Poděkování

Děkuji vedoucímu bakalářské práce Ing. Soběslavovi Valachovi za velice cenné a odborné rady při zpracování této bakalářské práce. Děkuji své rodině za podporu.

V Brně dne 8. června 2020 ……..

podpis autora

# Obsah

# Seznam obrázků

# Seznam tabulek



# Seznam rovnic

# 1 ÚVOD

Tato bakalářská práce se zabývá řízením elektricky komutovaných motorů za využití FPGA čipů a návrhem měniče. O danou problematiku se dlouhodobě zajímám. Řízení za využití FPGA čipu jsem zvolil, protože nabízí jiné prostředky než mikroprocesory, pro které nacházím využití. Náplň této práce je velmi různorodá a zahrnuje simulace, tvorbu softwaru i hardwaru. Měnič navrhuji pro malý jednoampérový BLDC motor a je mou snahou provést návrh tak, aby byl nezávislý na typu řízení. V případě potřeby použití pro řízení výkonnějších motorů by bylo nutné upravit hardware pro zvládnutí vyšších proudů.

V této bakalářské práci bude na začátku proveden průzkum typů řízení. Zvolený typ řízení bude realizován simulací a v případě, že se ukáže jako vhodný, bude implementován do FPGA. Navrhnutý měnič umožní na závěr ověření na skutečném BLDC motoru.

# 2 ROZBOR ZADÁNÍ

## 2.1 Seznámit se s možnostmi řízení BLDC motorů a jejich parametry

Jedná se o teoreticky založenou část, kterou dokumentuje kapitola „Typy řízení motorů“. Parametry motorů jsou využívány v průběhu celé práce.

## 2.2 Navrhnout a ověřit simulací vhodný model řízení

Byl zvolen typ řízení FOC. Jeho realizaci a výsledky popisují kapitoly „Realizace bloků Simulinku“ a kapitola „Výsledky simulací“.

## 2.3 Navrhnout schéma zapojení řídícího modulu

Celkově byly realizovány dva řídící moduly. Od prvního bylo upuštěno pro znatelné nevýhody vůči druhému. Návrh schéma popisuje kapitola „Návrh schéma zapojení driveru s DRV8305“.

## 2.4 Připravit návrh desky plošného spoje řídícího modulu

Návrh desky plošného spoje je popsán v kapitole „Návrh plošného spoje driveru s DRV8305“.

## 2.5 Koncepce řízení BLDC motoru v FPGA

Práce obsahuje dva možné způsoby implementace řízení do FPGA. První způsob popsaný v kapitole „Generování VHDL kódu“ objasňuje automatické generování HDL kódu z vytvořené simulace pro implementaci FOC řízení, které se ukázalo po získání více znalostí vhodné spíše pro PMSM motory. Druhý způsob popsaný kapitolou „Řídící program šestikrokové komutace“ implementuje klasické šestikrokové řízení používané pro BLDC motory.

## 2.6 Návrh metod verifikace funkce a zvolené koncepce

Při vývoji řídícího algoritmu probíhaly simulace v softwarovém nástroji MATLAB a prostředí Simulink. Pro FPGA je využíván software s obchodním názvem Vivado a zahrnuje nástroje pro syntézu a implementaci hardwarového popisu. Při oživování

plošného spoje probíhaly měření za použití osciloskopu. Práce se zabývá řízením motoru. Točí-li se motor podle požadavků, byla zvolená koncepce správná.

# 3 TYPY MOTORŮ [23]

BLDC i PMSM jsou synchronní motory. U synchronních motorů se magnetické pole statoru snaží dohnat magnetické pole rotoru. Největší točivý moment vzniká, když je magnetické pole statoru napřed o 90°. Naopak nulový točivý moment nastane, když se magnetické pole statoru a rotoru srovnají. Tento stav je nežádoucí.

## 3.1 BLDC motor

Tento typ motoru nemá komutátor. Skládá se ze statoru, na kterém je navinuté statorové vinutí a rotoru, který je tvořen permanentními magnety. Motor patří do skupiny EC motorů. Rozložení magnetů zajišťuje trapézový průběh zpětné elektromotorické napětí. Poloha BLDC motoru může být snímána halovými sondami. Tato semestrální práce je zaměřena na metodu řízení, která nevyužívá senzory. Poloha motoru je odhadovaná na základě měřených statorových proudů a napětí, nebo na základě měření zpětného elektromotorického napětí.

BLDC motory lze řídit tzv. šestikrokovou komutací, kdy měnič nabývá šesti kombinací, jež odpovídají šesti možným polohám (vektorům) motoru po šedesáti stupních. Tento typ řízení způsobuje, že motor je při provozu hlučnější, a to právě proto, že máme k dispozici pouhých šest pozic, kterými můžeme motor řídit. Vyšší hluk také způsobují vyšší harmonické složky obsažené v trapézovém průběhu.

Výhody BLDC motoru:

- Rotor nemá vinutí => není potřeba komutátor => nedochází k opalování kartáčů při předávání napájení na rotor => delší životnost
- Velice dobrý poměr výkon/hmotnost

Nevýhody BLDC motoru

- Cena permanentních magnetů rotoru způsobuje vyšší cenu těchto motorů
- Potřeba složitých a komplexních elektronických kontrolérů

## 3.2 PMSP motor

Narozdíl od BLDC motoru je PMSP motor řízen střídavým napětím. Stejně jako BLDC má na rotoru permanentní magnety. Má sinusové rozdělení magnetického pole rotoru.

# 4 TYPY ŘÍZENÍ MOTORŮ [12]

## 4.1 Skalární řízení

Při skalárním řízení dochází pouze k řízení za využití změny amplitudy a střídy PWM signálu přiváděného na měnič.

### 4.1.1 Šestikroková komutace

Jedná se o jednu z možných variant skalárního řízení. Fáze jsou aktivovány po šedesáti stupních. Při tomto typu řízení proud vždy protéká pouze dvěma fázemi, čímž lze dosáhnout přesně šesti možných variant během jedné otáčky. Není nutné znát polohu rotoru a řízení lze provádět v otevřené smyčce. Tento způsob se používá především pro roztočení motoru.

Řízení lze provést i za použití senzorů polohy rotoru. Polohu rotoru lze získat například použitím halových sond, enkodérů, resolverů a také pomocí takzvaných bez-senzorových metod. Výhodou je nízká výpočetní náročnost a poměrně jednoduchá realizace. Nevýhody oproti ostatním metodám jsou:

- Jedna z fází je vždy ve stavu vysoké impedance, za účelem měření zpětného elektromagnetického napětí. Aktivní jsou pouze dvě fáze a tím je dosaženo menšího točivého momentu
- Pouhých 6 kroků na otáčku způsobuje zvlnění průběhu točivého momentu
- Zvlnění dále způsobuje větší hlasitost motoru, vibrace a kratší životnost
- Regulace při nižších otáčkách je neefektivní a nepřesná

### 4.1.2 Sinusová komutace

Tento typ komutace je určen pro PMSM motory. Výhody sinusové komutace jsou menší zvlnění točivého momentu a přesnější řízení oproti šestikrokové komutaci.

Nevýhoda sinusové komutace je složitý řídící algoritmus, který vyžaduje přesnou informaci o poloze motoru.

## 4.2 Vektorového řízení [14] [15]

Vektorové řízení se skládá z řady **matematických transformací** a komplexní modulační metody, tzv. **SVM modulace**, poskytující vysokou účinnost a točivý moment. Vektorové řízení se používá pro ovládání **synchronních motorů**.

Pro realizaci vektorového řízení potřebujeme znát **relativní polohu rotoru ke statoru**. V případě snímání špatné polohy by PI regulátory pracovaly se špatnými daty. Polohu motoru můžeme měřit stejnými způsoby jako u skalárního řízení senzorově. Bez-senzorové řízení je při použití této metody náročnější, jelikož

vyžaduje tzv. sledovač elektromotorického napětí, což je matematický model motoru simulující elektromotorické napětí při znalosti napětí a proudu skutečného motoru.

Vektorový součet proudů představuje rotující vektor magnetického pole statoru. Pro regulaci potřebujeme naměřené proudy pomocí transformací převést na vhodný formát. Jak je známo z matematiky, bod v prostoru lze vyjádřit za použití pouhých 2 lineárně nezávislých vektorů. Této transformaci se říká **Clarkova transformace** a je obvykle označována pod názvem **abc to αβ**.

Aplikujeme-li tuto transformaci na měřené proudy, dostaneme místo tří veličin pouze dvě. Tyto veličiny se však stále nehodící se pro regulaci podle klasické teorie řízení. Jsou totiž proměnné s natočením motoru. Jsou v tzv. **rotujícím rámci**.

Aby šla regulace realizovat, je potřeba udělat transformaci do **stacionárního rámce**. Tuto transformaci nazýváme **Parkova transformace**. Můžeme se setkat také s označením **αβ to dq**, jež vyplývá z názvů vstupů a výstupů**.** Tato transformace zahrnuje násobení rotujícího vektoru úhlem magnetického toku. Veličiny **αβ** se tím převedou na stejnosměrné **dq** hodnoty, jež jsou konstantní s natočením motoru. Představují **magnetický tok** a **točivý moment**.

Regulací d (direct) a q (quadrature) os získáváme velice přesné a účinné řízení. Převedení do stacionárního rámce nám umožňuje regulaci rychlosti motoru v daleko větším rozsahu než při skalárním řízení.

Jelikož **d** i **q** jsou funkcí proudu, navzájem se ovlivňují. Například pokud zvýšíme točivý moment zvýšením frekvence, magnetický tok poklesne.

Veličina (d) neposkytuje užitečný točivý moment a pouze zvětšuje opotřebení ložiskového uložení motoru. Veličinu d regulujeme na nulovou hodnotu.

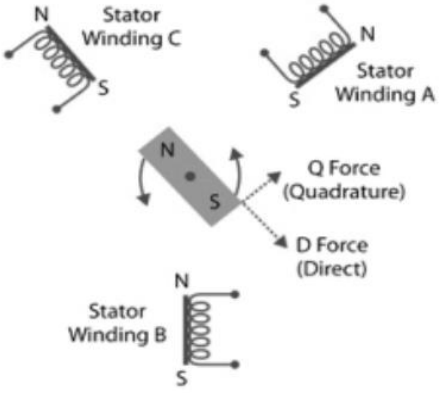
Veličina (q) představuje užitečný točivý moment. Regulujeme ji v závislosti na tom, zda potřebujeme zrychlit či zpomalit otáčky motoru. Při dosažení konstantních otáček je veličina q regulována na nulovou hodnotu. Naopak při potřebě snížit či zvýšit otáčky je regulována do kladných či záporných hodnot.

Vektorové řízení umožnuje mít v každém momentě natočení **magnetického pole statoru** kolmo na **magnetické pole permanentních magnetů rotoru**. Touto vlastností získáváme maximální točivý moment a velmi přesné řízení.

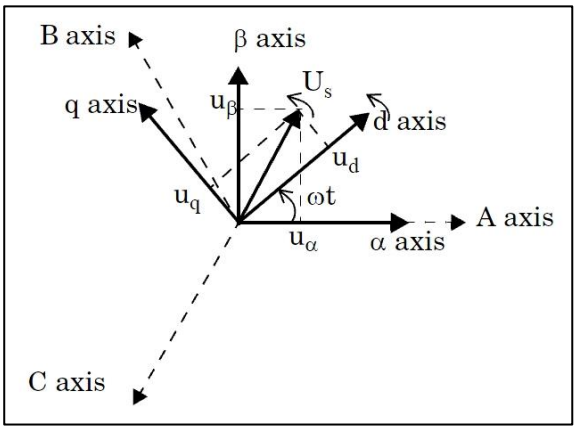
Na Obrázek 3 můžeme vidět graficky znázorněno pořadí transformací a průběhy signálů před a po každé transformaci.

Na Obrázek 2 můžeme vidět polohu jednotlivých veličin v kartézské soustavě. Lze vidět polohu měřených vektorů proudu A, B, C, poté vidíme veličiny **α** a **β** po Clarkově transformaci, a nakonec i veličiny **D** a **Q** po Parkově transformaci.

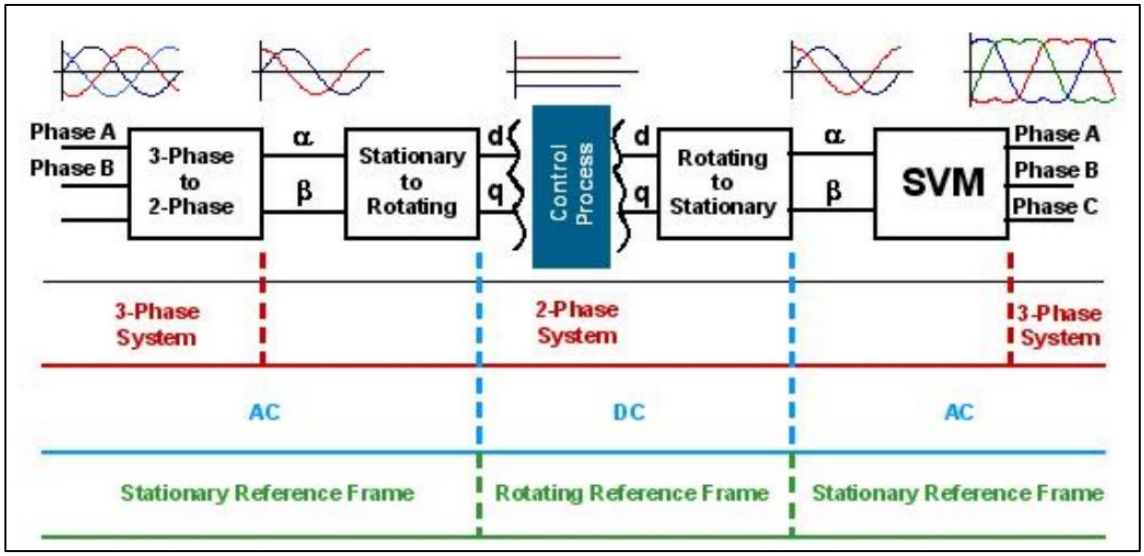
Na Obrázek 1 můžeme vidět polohu **D** a **Q** veličin v závislosti k permanentnímu magnetu rotoru




**Obrázek 1 Zobrazení DQ os v souvislosti s magnetem [12]**




**Obrázek 2 Zobrazení veličin v kartézské soustavě [6]**




**Obrázek 3 Grafické znázornění propojení transformací [12]**

## 4.2.1 ABC to DQ transformace [6]

Pro vektorové řízení není potřeba dělat Clarkovu a Parkovu transformaci zvlášť. Model tedy používá tzv. Clark-Park transformaci, která se chová stejně.

Používají se dvě konvence, kdy:

- Rotující rámec je zarovnaný s osou A v čase t = 0
- Rotující rámec je zarovnaný o 90° za osou A v čase t = 0

$$U_s = u_d + j \cdot u_q = (u_\alpha + j \cdot u_\beta) \cdot e^{-j\left(\omega t - \frac{\pi}{2}\right)}$$

$$\begin{bmatrix} u_d \\ u_q \\ u_0 \end{bmatrix} = \frac{2}{3} \begin{bmatrix} \sin(\omega t) & \sin\left(\omega t - \frac{2\pi}{3}\right) & \sin\left(\omega t + \frac{2\pi}{3}\right) \\ \cos(\omega t) & \cos\left(\omega t - \frac{2\pi}{3}\right) & \cos\left(\omega t + \frac{2\pi}{3}\right) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} u_a \\ u_b \\ u_c \end{bmatrix}$$

**Rovnice 4.2-1 [6]**

### 4.2.2 DQ to AlphaBeta transformace [6]

Zpětnou transformaci stačí provést jako zpětnou Parkovu transformaci, jelikož pro následující blok SVM je jednodušší počítat délku rotujícího vektoru jen ze dvou vektorů, namísto tří.

$$u_\alpha + j \cdot u_\beta = (u_d + j \cdot u_q) \cdot e^{j\omega t}$$

$$\begin{bmatrix} u_\alpha \\ u_\beta \\ u_0 \end{bmatrix} = \begin{bmatrix} \cos(\omega t) & -\sin(\omega t) & 0 \\ \sin(\omega t) & \cos(\omega t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_d \\ uq \\ u_0 \end{bmatrix}$$

**Rovnice 4.2-2 [7]**

### 4.2.3 Regulátor [20][21][22]

Pro řízení motorů se používá kaskádní regulace. Vnitřní smyčka reguluje proud motorem. Nadřazená regulační smyčka reguluje rychlost a v případě potřeby lze přidat další nadřazenou smyčku pro regulaci polohy.

Pro regulaci D a Q proudů se používá PI regulátor. U řízení motoru je požadavek, aby se nula regulátoru vyrušila s pólem motoru. Jediný pól motoru vzniká ve vinutí motoru a je rovný jeho časové konstantě. Pro regulaci je vhodné použít sériový PI regulátor, kde změna proporcionální složky neposouvá nulu regulátoru. [4]

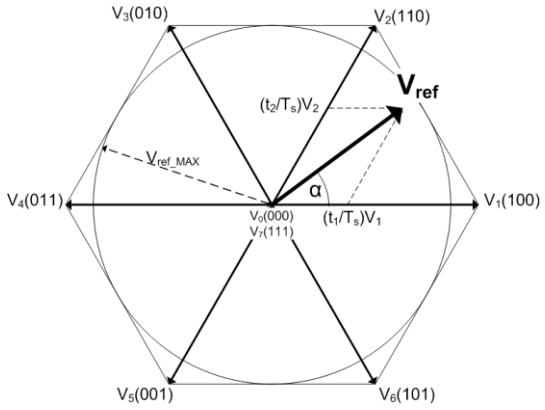
### 4.2.4 SVM – Space Vector Modulation [13][5][8]

BLDC motor má 3 vinutí navzájem posunuté o 120 stupňů. Napájíme-li motor stejnosměrným napětím, můžeme dosáhnout šesti aktivních stavů, jež jsou od sebe posunuté o 60 stupňů, a dvou pasivních stavů. Tyto stavy popisuje tabulka 3.2-1.

SVM je modulační algoritmus, který má na výstupu tři pulzně šířkově modulované signály, které mohou vytvořit vektor o jakémkoliv natočení tím, že vždy přepínají pouze mezi dvěma kombinacemi stavů (například 100 a 110). Stavy lze vidět na Obrázek 4. Polohu vektoru mezi těmito dvěma stavy určuje poměr dob setrvání v jednotlivých stavech.

Kromě natočení vektoru lze ovlivnit i jeho velikost, a to tím, že navíc přepínáme i mezi nulovými stavy. Při spínání vznikají ztráty a je tedy vhodné přepínat na ten nulový vektor, na jehož přepnutí z aktivního stavu se provede méně přepnutí.

Pulzně modulované signály vytvořené SVM jsou přiváděny na měnič, který má šest tranzistorů. Spodní tranzistory měniče mají vždy opačnou řídící hodnotu než horní tranzistory.




**Obrázek 4 Grafické znázornění vektorů [13]**

**Tabulka 1 Stavy tranzistorů pro jednotlivé vektory [13]**

| Vector | $+A$ | $+B$ | $+C$ | $-A$ | $-B$ | $-C$ | $V_{AB}$ | $V_{BC}$ | $V_{CA}$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| $V_0 = 000$ | OFF | OFF | OFF | ON | ON | ON | 0 | 0 | 0 | Zero Vector |
| $V_1 = 100$ | ON | OFF | OFF | OFF | ON | ON | $+V_{DC}$ | 0 | $-V_{DC}$ | Active Vector |
| $V_2 = 110$ | ON | ON | OFF | OFF | OFF | ON | 0 | $+V_{DC}$ | $-V_{DC}$ | Active Vector |
| $V_3 = 010$ | OFF | ON | OFF | ON | OFF | ON | $-V_{DC}$ | $+V_{DC}$ | 0 | Active Vector |
| $V_4 = 011$ | OFF | ON | ON | ON | OFF | OFF | $-V_{DC}$ | 0 | $+V_{DC}$ | Active Vector |
| $V_5 = 001$ | OFF | OFF | ON | ON | ON | OFF | 0 | $-V_{DC}$ | $+V_{DC}$ | Active Vector |
| $V_6 = 101$ | ON | OFF | ON | OFF | ON | OFF | $+V_{DC}$ | $-V_{DC}$ | 0 | Active Vector |
| $V_7 = 111$ | ON | ON | ON | OFF | OFF | OFF | 0 | 0 | 0 | Zero Vector |

### 4.2.5 Sledovač zpětného elektromotorického napětí

Pro Parkovu transformaci je potřeba znát úhel natočení vektoru magnetického pole. V případě senzorového řízení, při použití například halových sond pro měření polohy motoru, lze tento úhel určit snadno.

Jelikož používáme bez-senzorové řízení, je potřeba tento úhel odhadnout za pomocí matematického modelu stejnosměrného motoru. Tento model pracuje s měřenými proudy a napětími na fázích motoru.

Určení polohy přímým měřením zpětného elektromotorického napětí je znemožněno, protože SVM modulace má vždy aktivní všechny tři fáze.

# 5 KOMPONENTY PRO HW REALIZACI

## 5.1 Gate driver

Výkonové MOSFET tranzistory měniče potřebují při spínání dostatečný proud do řídícího hradla. Potřebný proud je větší než proud, který může dodat FPGA samotné. Je potřeba použít tzv. gate driver, který obvykle integruje nábojovou pumpu pro dodání většího proudu na změnu stavu hradla tranzistoru. Tato součástka je ovládaná výstupy z FPGA desky a řídí hradla výkonových tranzistorů.

## 5.2 Analogově digitální převodník

Pro měření proudu větve měniče se používá rezistor s velmi nízkým odporem. Úbytek napětí na odporu lze přepočítat na proud větví měniče. Jelikož je napětí kvůli malým hodnotám odporů velmi malé, je vhodné použít zesilovač, který napětí zesílí na úroveň vhodnou pro další zpracování. Pro určení hodnoty rezistoru pro měření proudu a zesílení zesilovače platí základní pravidla:

- Čím vyšší odpor rezistoru, tím větší úbytek napětí, čímž se získá lepší rozlišení snímaného proudu a nároky na kvalitu zesilovače klesají
- Velikost odporu nemůže být příliš velká, jelikož s velikostí odporu stoupá velikost ztrátového výkonu
- Větší zesílení zesilovače umožňuje použít rezistor s menším odporem, ale za cenu zvýšení šumu

Řídící model pracuje na základě přesných dat o proudu a napětí na fázích motoru.

Proud fázemi měříme nepřímo měřením úbytku napětí na rezistorech. Toto napětí je zesíleno zesilovači a je potřeba ho převést z analogové hodnoty na digitální.

Napětí měříme přímo přes odporový dělič, připojený přímo na fázi motoru. Na analogově digitální převodník jsou kladeny nároky, které určuje typ aplikace. Aby byl akční zásah vektorového řízení přesně spočítaný, je zapotřebí vzorkovat úbytky napětí rezistorů a napětí na napěťových děličích na všech fázích ve stejný čas. AD převodníky desky Basys 3 nepodporují současné vzorkování, proto je potřeba použít externí AD převodník.

Další požadavek je, aby vzorkování průběhu proudu bylo synchronizované s frekvencí SVM PWM. Tím se omezí efekt PWM na vzorkovaný průběh.

Analogově digitálních převodníků existuje více typů, z nichž se pro řízení motorů používají především dva druhy. Konkrétně **převodníky s postupnou aproximací** a **převodníky typu sigma-delta**.

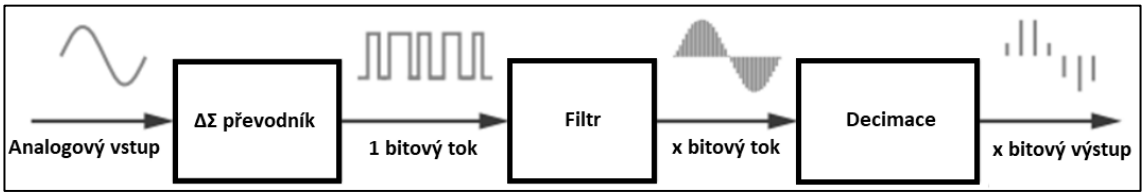
## 5.3 DELTA-SIGMA převodník [9]

Delta sigma převodník vzorkuje signál na frekvenci násobně větší (obvykle 64x větší), než výsledná frekvence. Převádí vstupní signál na **1bitový datový tok**. Při vzorkování dochází k **posunu kvantizačního šumu do vyšší frekvenční oblasti**.

Datový tok je zpracováván filtrem, který převede jednobitový signál na vícebitový a **odstraní kvantizační šum z vyšších frekvencí**.

Následuje **decimace**, což je proces snižující vzorkovací frekvenci směrem dolů, tak, aby odpovídala frekvenci řídícího systému. Na obr. 4.2-1 můžeme vidět grafické znázornění výše popsaného průběhu. Filtrování a decimace se provádí použitím tzv. sync filtru (ideálně filtr, jež odstraní všechny složky nad určitou frekvencí, bez toho, aby ovlivnil spodní frekvence).

Tento typ převodníku poskytuje kvalitněji zpracovaný signál obvykle s vyšším rozlišením. Jeho nevýhodou je celková doba zpracování. Tedy doba od začátku do konce zpracování signálu.




**Obrázek 5 Grafické znázornění bloků delta-sigma převodníku**

## 5.4 Převodník s postupnou aproximací

Převodník s postupnou aproximací se skládá z n-bitového registru, jehož hodnota je převáděna digitálně analogovým převodníkem na analogovou hodnotu a porovnávána s převáděným signálem za pomocí komparátoru. Jednotlivé bity n-bitového registru jsou nastavovány od nejvyššího váhového bitu (MSB).

## 5.5 SPI rozhraní AD převodníků

SPI rozhraní je rozhraní určené pro rychlé datové přenosy na vzdálenost menší než jeden metr. SPI rozhraní má čtyři vodiče CLK, MOSI, MISO a ChipSelect. Běžně pracuje na frekvenci 25MHz a může být realizováno ve více variantách, které se liší v polaritě hodinového signálu a v tom, zda jsou data čtená při náběžné či sestupné hraně hodinového signálu. Je tedy zapotřební zjistit, jakou z těchto 4 variant možného nastavení používá AD převodník.

# 6 IMPLEMENTACE UART [24]

Jedná se o běžný protokol, dostupný v mnoha variantách. Implementace tedy nespočívá ve vytváření vrstvy, jež provádí samotnou serializaci, ale v definování konkrétního datového rámce pro naši aplikaci. Na straně osobního počítače lze data jednoduše číst použitím terminálu, a v případě komplexnějšího systému je vhodné vytvořit aplikaci pracující s daty, případně poskytující vizualizaci dat.

## 6.1 Datový formát přenášených dat

Data jsou posílána přes sériovou linku jako ASCII znaky. Tento způsob přenosu je dostačující při malém datovém toku. Bude-li potřeba přenášet větší množství dat, je nutné upravit protokol tak, aby pracoval s binárními údaji.

## 6.2 C# Aplikace [17]

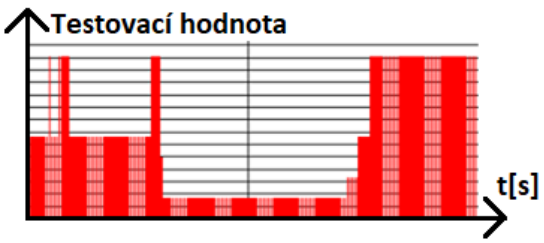
Aplikace byla vytvořena za účelem automatizace a zjednodušení příjmu a odesílání dat. Současná verze aplikace umožňuje posílat čísla v rozsahu 0 až 65 535 do desky Basys 3. Při příjmu dat jsou data kromě zobrazení numerické hodnoty vykreslována do grafu v závislosti na čase.

## 6.3 Testování komponenty přijímače

Sériový přijímač byl nastaven tak, že přijímá jednu testovací hodnotu. Datový řetězec lze se současnou implementací rozšířit na téměř libovolnou délku. Správnost přijatých dat je ověřována na displeji Basys 3. Pošleme-li výše zmíněnou aplikací hodnotu 4363, zobrazí se na sedmi segmentovém displeji stejná hodnota.

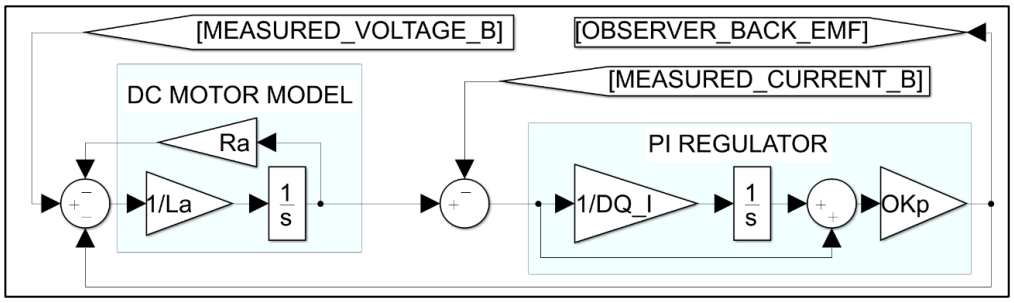
## 6.4 Testování komponenty vysílače

Přenos byl otestován tak, že hodnota pro odeslání je nastavena na přepínačích desky Basys 3 v binárním formátu. Tato hodnota je následně za použití **binary_bcd** komponenty převedena na pět BCD čísel. Vysílač má implementovanou lookup tabulku, pomocí které překládá BCD čísla na ASCII znaky, jejichž binární hodnotu postupně vysouvá za využití shift registru do přijímače (v tomto případě počítač).




**Obrázek 6 Ukázka části grafu z C# aplikace**

# 7 SLEDOVAČ ZPĚTNÉHO ELEKTROMOTORICKÉHO NAPĚTÍ [18]

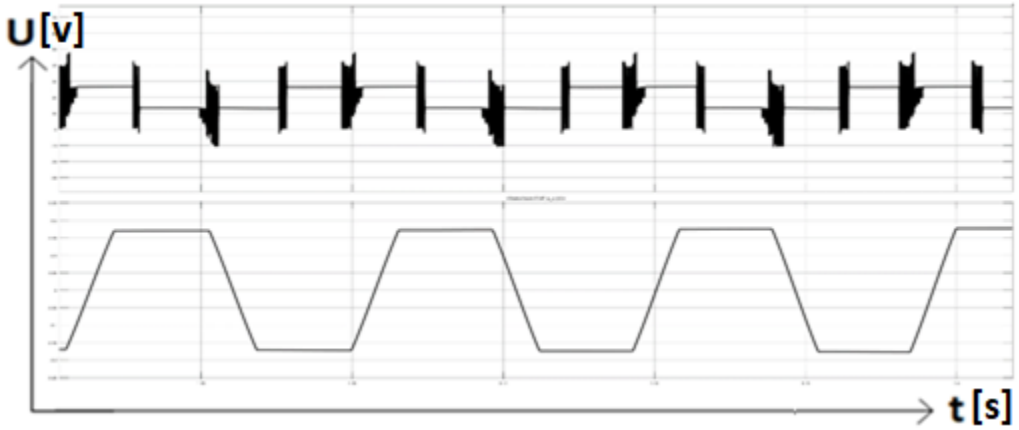



**Obrázek 7 Zapojení sledovače el. mot. napětí**

Pro určení polohy motoru potřebujeme znát zpětné elektromotorické napětí. Zpětné elektromotorické napětí, jak název napovídá, působí opačným směrem proti napětí aplikovanému na motor. Není tedy přímo měřitelné, ale je obsaženo v napětí, které měříme na jednotlivých fázích. Zpětné elektromotorické napětí lze měřit při použití vhodné PWM, která nechává jednu fázi připojenou na stav vysoké impedance, avšak SVM PWM má vždy aktivní všechny fáze.

Pro jeho získání je potřeba sestavit model stejnosměrného motoru (Obrázek 7), na který přivádíme hodnotu napětí z jedné fáze řízeného motoru. Výstupem modelu stejnosměrného motoru je proud. Vypočítaný proud je rozdílný od měřeného proudu, protože na měřený proud již působí zpětné elektromotorické napětí.

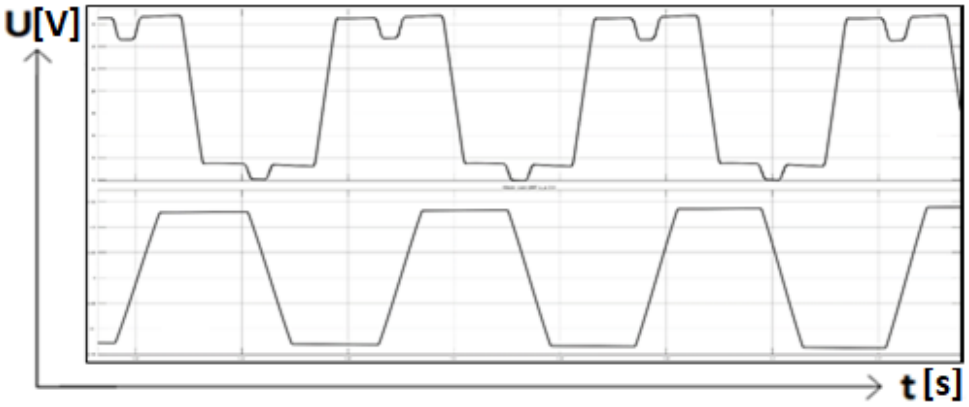
Rozdíl měřeného a počítaného proudu je chyba, kterou přivádíme na PI regulátor. Budeme-li výstup regulátoru zpětnou vazbou odečítat od přiváděného napětí na model, regulátor bude vstupní napětí modelu regulovat tak, aby proudy byly totožné. Nyní je výstup regulátoru hledané zpětné elektromotorické napětí.



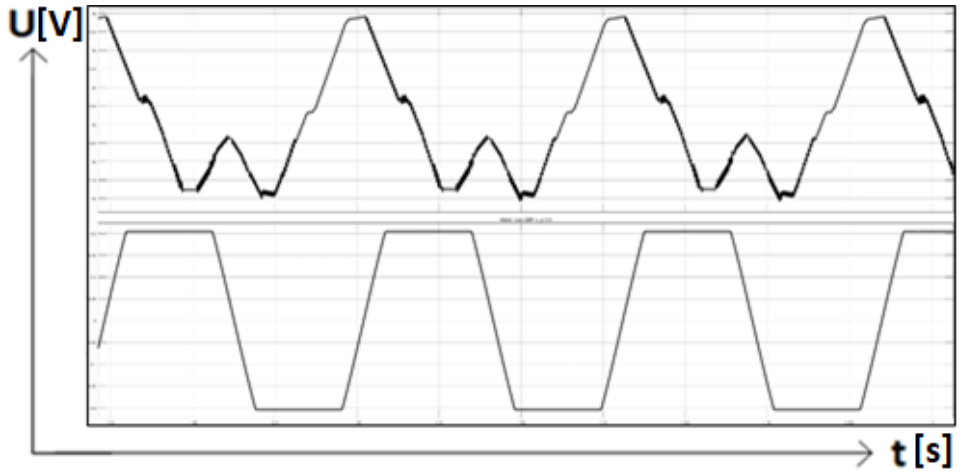

**Obrázek 8 Průběh zpětného elektromotorického napětí v čase**

Na Obrázek 8 lze vidět, že bez jakéhokoliv filtrování obsahuje zákmity. Byl aplikován filtr průměrující posledních 200 vzorků. Po vyfiltrování je průběh

znatelně čitelnější (Obrázek 9). Všechny vzorky byly doposud zaznamenány v grafech během změny rychlosti, kdy regulátor reguluje na konkrétní rychlost, ale zatím jí nedosáhl. Tedy vytváří točivý moment. Jakmile se rychlost ustálí, tak zpětné elektromotorické napětí ztratí předchozí tvar (Obrázek 10).

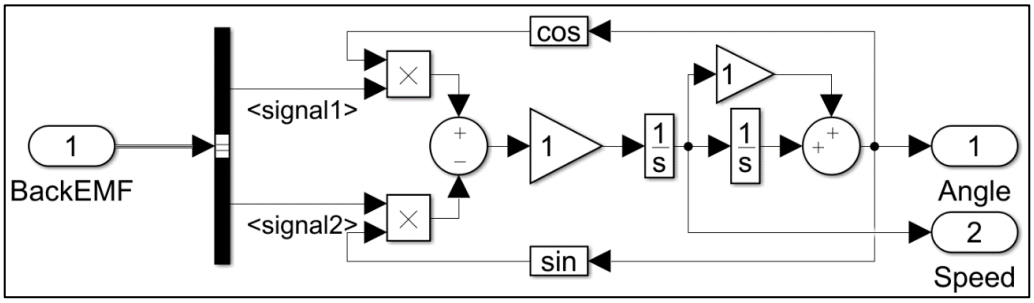



**Obrázek 9 Průběh z obr. 8 s přidanou filtrací**




**Obrázek 10 Průběh reálného a ideálního BackEMF**

Na Obrázek 11 lze vidět zapojení tzv. Angle demodulator [4], který se používá pro výpočet úhlu ze zpětného elektromotorického napětí. V současnosti však nefunguje ani s ideálním průběhem zpětného elektromotorického napětí z modelu BLDC motoru.




**Obrázek 11 Zapojení demodulátoru úhlu**

# 8 REALIZACE BLOKŮ SIMULINKU

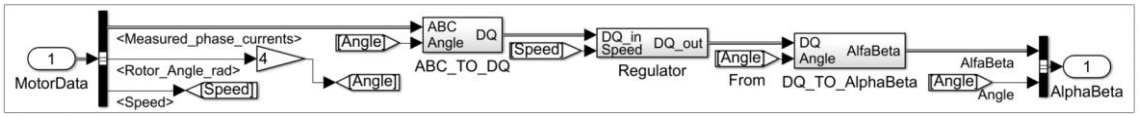
Model řízení je rozdělen na dvě části. První část je regulační, druhá modulační. Při tvoření Simulinkového modelu bylo vyhodnoceno jako výhodné používat blok subsystému, který umožňuje zapouzdřit jednotlivé části a model se stává přehlednější. Model byl rozdělen do subsystémů tak, aby jeden subsystém vždy obsahoval jen jednu konkrétní část z řetězce úloh, které je potřeba pro řízení motoru provést. V Simulinku máme dvě možnosti modelování rovnic:

- Použití bloku **Function** – kde rovnici zapíšeme v matematickém formátu
- Sestavení rovnice z jednotlivých bloků primitivních matematických operací

První možnost je nevyhovující, jelikož z bloku Function nelze generovat HDL kód. Bylo by také znemožněno sledování maximální a minimální hodnoty na výstupu každé části matematické operace, což by ztížilo převod do pevné řádové čárky.

## 8.1 Regulační část

Implementuje transformace a regulátory potřebné pro řízení motoru. Na Obrázek 12 lze vidět blokové schéma celé regulační části.



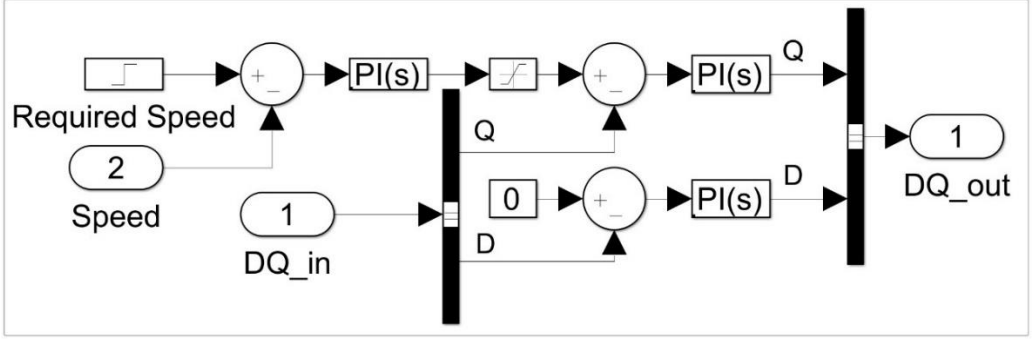

**Obrázek 12 Blokové schéma regulační části**

### 8.1.1 Blok ABC_TO_DQ

Tento blok implementuje Clarkovu a Parkovu transformaci. Vstupy bloku jsou měřené proudy v jednotlivých větvích měniče a úhel natočení magnetického pole (Z modelu nebo bloku Angle_observer).

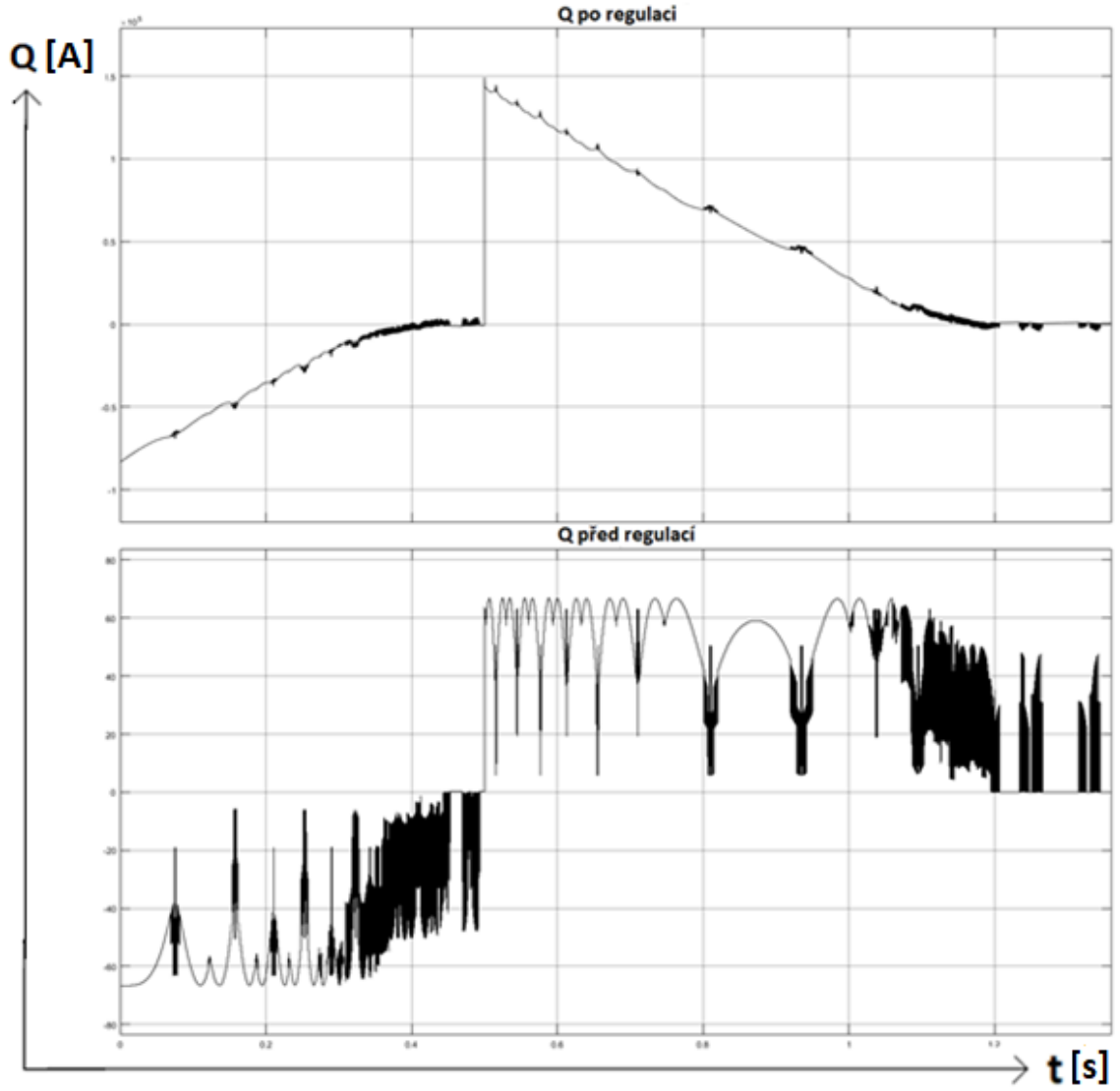
Výstupem tohoto bloku je proud rozdělený na dvě samostatné stejnosměrné složky $I_D$ a $I_Q$, představující **magnetický tok** a **točivý moment.**

### 8.1.2 Regulátor




**Obrázek 13 Kaskádní zapojení PI regulátorů**

Na Obrázek 13 vidíme kaskádní zapojení pro regulaci rychlosti. Skládá se ze dvou PI regulátorů proudu ve vnitřní smyčce a jednoho PI regulátoru ve vnější smyčce pro regulaci rychlosti. Jelikož je řídící model rozdělený na více bloků, nelze vidět zpětnou vazbu. Na Obrázek 14 je graficky znázorněna hodnota Q před a po regulaci v závislosti na čase. Na Obrázek 15 je průběh veličiny D.



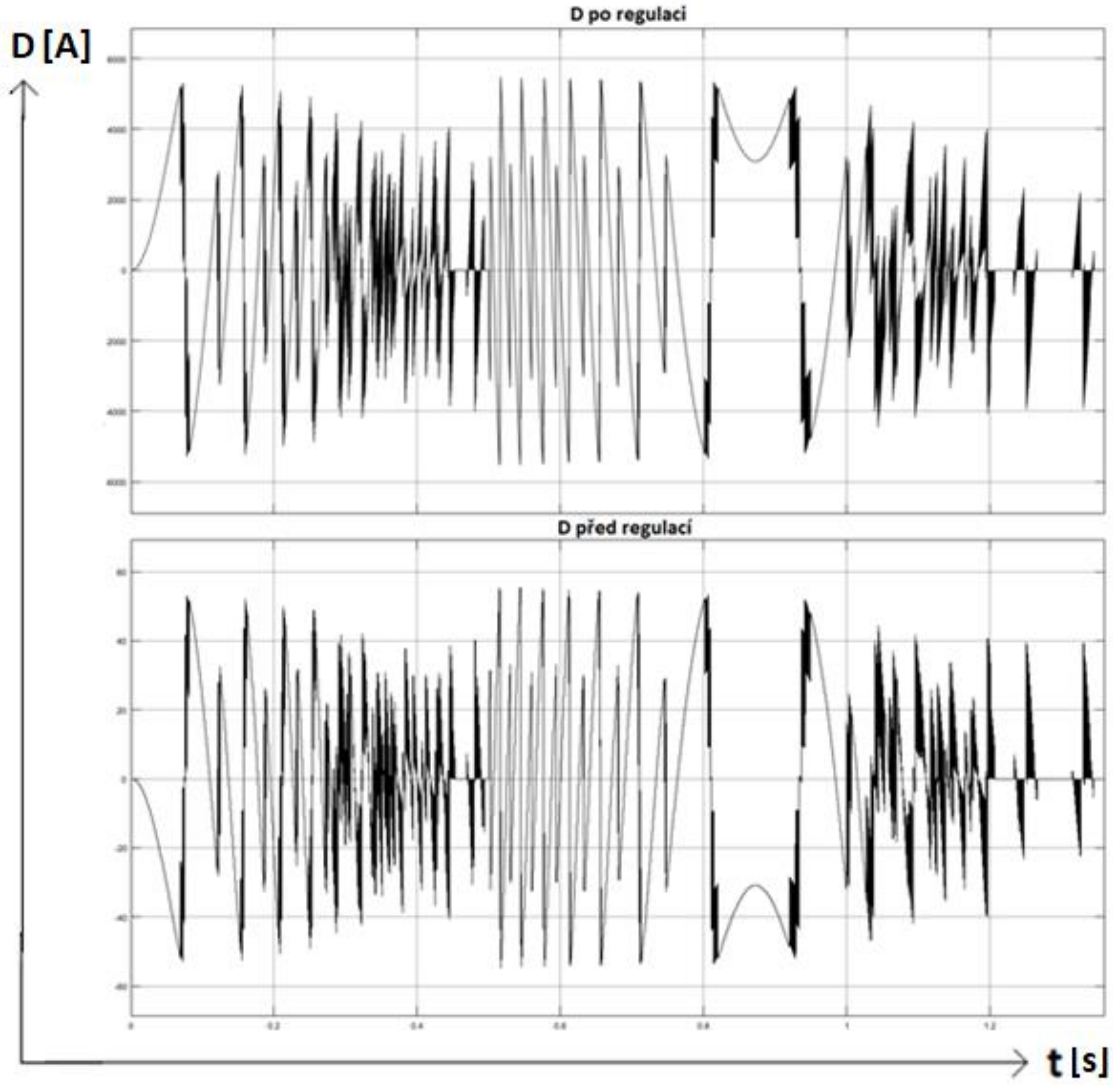

**Obrázek 14 Průběhy veličiny Q před a po regulaci v závislosti na čase**

Graf byl zachycen po regulaci rychlosti na -300 otáček na minutu, kdy bylo po jejich dosažení zahájeno regulování na +200 otáček za minutu. Hodnota Q představuje točivý moment a podle předpokladů je tedy při dosažení otáček blízká nule.

Hodnota D je vždy regulována na nulovou hodnotu. Lze si všimnout, jak je horní graf téměř zrcadlově obrácen oproti spodnímu grafu. Regulace probíhá podle předpokladů. Bylo regulováno na -300 otáček/min a poté +200 otáček/min. Těchto hodnot bylo dosaženo v časech t = 0,5 s a t = 1,2 s.

### 8.1.3 Blok Sine HDL Optimized

V modelu je často potřeba využít goniometrické funkce jako Sin a Cos. Tyto bloky však neumožňují generování HDL kódu a je potřeba vzít speciálně určený blok **Sine HDL Optimized**, který vygeneruje do paměti lookup tabulku s konkrétním počtem již vypočítaných hodnot. Počet těchto hodnot nastavuje proměnná spouštěcího scriptu.




**Obrázek 16 Průběhy veličiny D před a po regulaci v závislosti na čase**

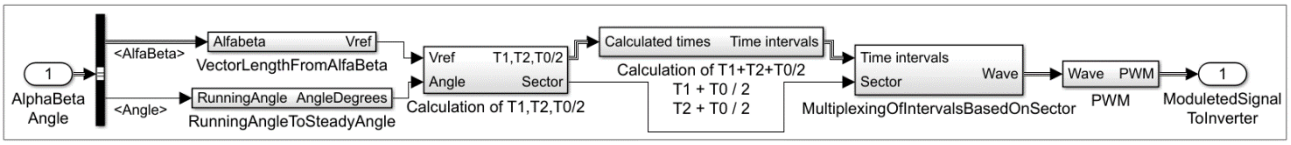
### 8.1.4 Integrátor

Stejně jako z bloků Sin a Cos, není možné generovat HDL kód ani z bloku **integrátoru**. Je nutné použít integrační blok určený pro diskrétní model, který generování HDL kódu podporuje. Blok integrace pro generování HDL umožňuje integraci dopřednou i zpětnou Eulerovou metodou.

## 8.2 Modulační část - SVM (Space Vector Modulation)

**Space vector modulation** je modulační technika, která převádí natočení a velikost požadovaného vektoru na šest pulzně šířkově modulovaných signálů, které jsou přiváděny na řídící hradla měniče.

SVM algoritmus požaduje úhel natočení, jež je stejný jako v předchozích blocích a délku vektoru, kterou spočítáme z Pythagorovy věty.




**Obrázek 17 Blokové schéma modulační části**

### 8.2.1 RunningAngleToSteadyAngle

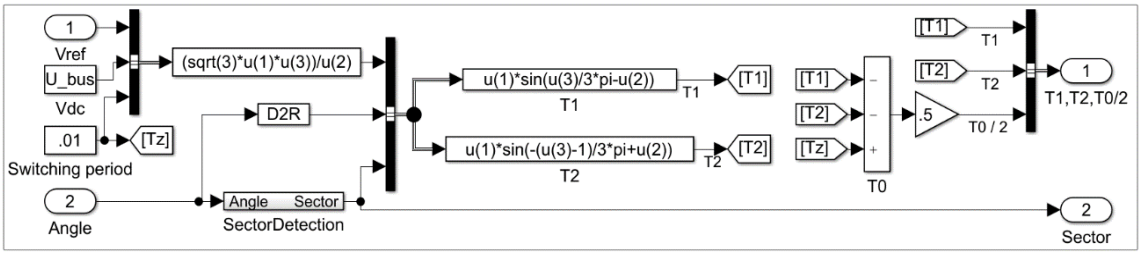
Blok slouží pro simulace, jež používají úhel poskytovaný simulinkovým modelem motoru, který se jmenuje Permanent Synchronous machine. Úhel není zobrazován od 0° do 360°, ale je stále zvyšován. Zvyšující se úhel je dělen hodnotou 360°. Zbytek z tohoto dělení je úhel v rozmezí 0° do 360°, což je zároveň výstup tohoto bloku.

### 8.2.2 VectorLengthFromAlfaBeta

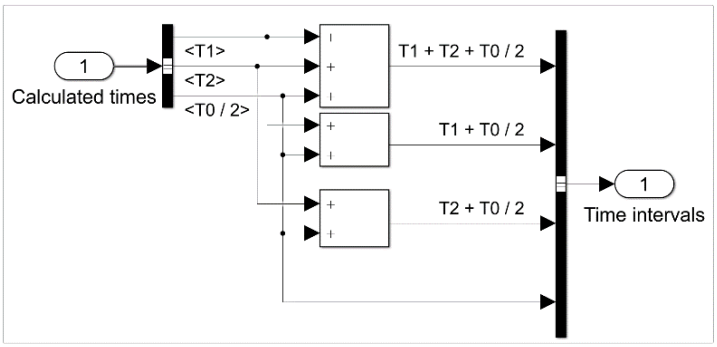
Výstupem regulační části jsou dva signály Alpha a Beta, které by šlo zpětnou Clark transformací převést zpět na původní 3 fáze. Vstupy Space vector modulace jsou však délka a požadovaný úhel natočení. Je tedy nejvýhodnější získat délku vektoru ze signálů Alpha a Beta.

### 8.2.3 Výpočet signálu pro generování SVM PWM

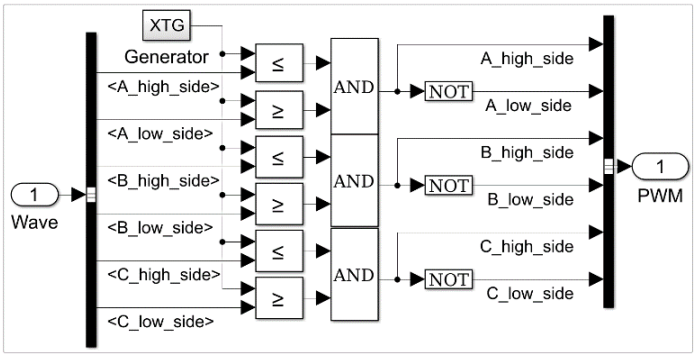
Pro vytvoření SVM PWM je potřeba provést řadu matematických operací. Matematické operace byly rozděleny do 3 bloků pro větší přehlednost. První blok můžeme vidět na Obrázek 18. Dochází zde k náročnějším výpočtům goniometrických funkcí. V dalším bloku (Obrázek 19) se signály pouze sčítají a v posledním bloku (Obrázek 20) dochází ke generování SVM PWM.




**Obrázek 18 Realizace výpočetní části SVM**




**Obrázek 19 Zapojení bloku CalculationOfIntervals**




**Obrázek 20 Zapojení PWM bloku SVM**

## MultiplexingOfIntervalsBasedOnSectors blok

V tomto bloku jsou předem spočítané 4 časové intervaly za použití bloku *Multiswitch* přepínány v závislosti na sektoru. Multiswitche přepínají vstupy na výstup podle tTabulka 2.

**Tabulka 2 Intervaly sepnutí pro jednotlivé sektory [8]**

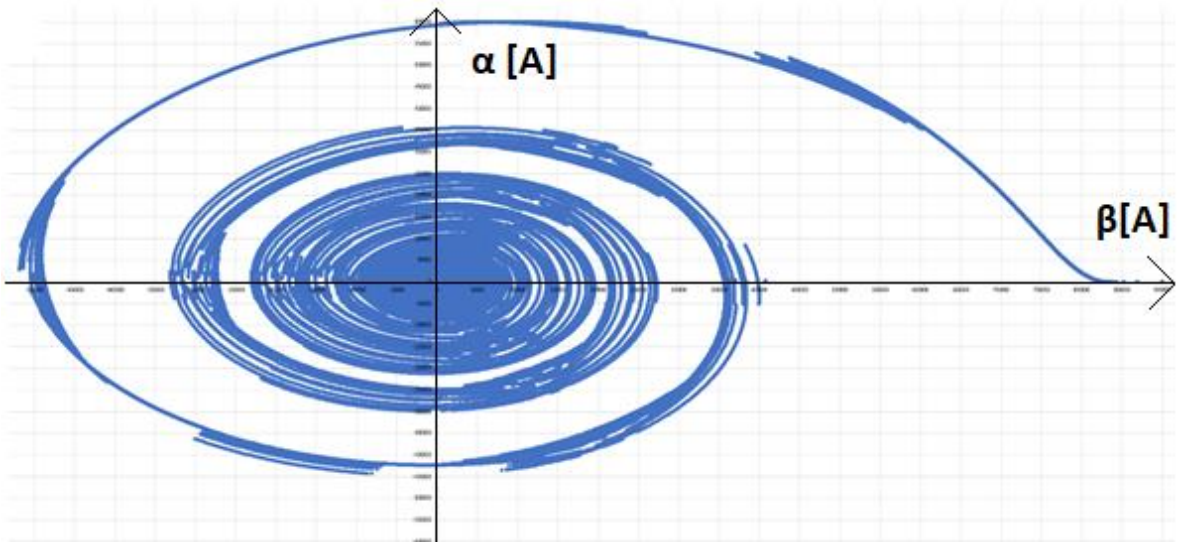
| Sektor | Horní tranzistory měniče | Sektor | Spodní tranzistory měniče |
|---|---|---|---|
| 1 (0°-60°) | $S_1 = T_1 + T_2 + T_0/2$ | 1 (0°-60°) | $S_2 = T_1 + T_2 + T_0/2$ |
| | $S_3 = T_2 + T_0/2$ | | $S_4 = T_0/2$ |
| | $S_5 = T_0/2$ | | $S_6 = T_1 + T_0/2$ |
| 2 (60°-120°) | $S_1 = T_1 + T_0/2$ | 2 (60°-120°) | $S_2 = T_1 + T_2 + T_0/2$ |
| | $S_3 = T_1 + T_2 + T_0/2$ | | $S_4 = T_2 + T_0/2$ |
| | $S_5 = T_0/2$ | | $S_6 = T_0/2$ |
| 3 (120°-180°) | $S_1 = T_0/2$ | 3 (120°-180°) | $S_2 = T_1 + T_0/2$ |
| | $S_3 = T_1 + T_2 + T_0/2$ | | $S_4 = T_1 + T_2 + T_0/2$ |
| | $S_5 = T_2 + T_0/2$ | | $S_6 = T_0/2$ |
| 4 (180°-240°) | $S_1 = T_0/2$ | 4 (180°-240°) | $S_2 = T_0/2$ |
| | $S_3 = T_1 + T_0/2$ | | $S_4 = T_1 + T_2 + T_0/2$ |
| | $S_5 = T_1 + T_2 + T_0/2$ | | $S_6 = T_2 + T_0/2$ |
| 5 (240°-300°) | $S_1 = T_2 + T_0/2$ | 5 (240°-300°) | $S_2 = T_0/2$ |
| | $S_3 = T_0/2$ | | $S_4 = T_1 + T_0/2$ |
| | $S_5 = T_1 + T_2 + T_0/2$ | | $S_6 = T_1 + T_2 + T_0/2$ |
| 6 (300°-360°) | $S_1 = T_1 + T_2 + T_0/2$ | 6 (300°-360°) | $S_2 = T_2 + T_0/2$ |
| | $S_3 = T_0/2$ | | $S_4 = T_0/2$ |
| | $S_5 = T_1 + T_0/2$ | | $S_6 = T_1 + T_2 + T_0/2$ |

## 8.2.4 PWM blok

Současná implementace SVM PWM disponuje vlastností, kterou je důležité zmínit. Při modulaci dochází k porovnávání **vytvořených signálů** a **trojúhelníkového průběhu**. Velikost točivého momentu se odvíjí od poměru velikostí těchto dvou veličin. Velikost generovaných průběhů je závislá na rozdílu požadované a aktuální rychlosti.

# 9 VÝSLEDKY SIMULACÍ

## 9.1 Rotující vektor




**Obrázek 22 Zobrazení koncového bodu rotujícího vektoru**

Čára na Obrázek 22 představuje koncový bod rotujícího vektoru. Motor je regulován na určitou rychlost. Čím více se k dané rychlosti blíží, tím se čára na grafu blíží ke středu. Čím větší je vzdálenost od středu (čím větší je délka rotujícího vektoru), tím větší je točivý moment. Lze tedy vypozorovat závislost mezi odchylkou rychlosti a točivým momentem. Při klesajícím rozdílu požadované a aktuální rychlosti klesá točivý moment.

V tomto grafu se po celou dobu nacházíme v oblasti, kdy jsme blízko regulované rychlosti a SVM modulace již využívá nulové vektory. V případě velkého rozdílu rychlostí by SVM modulace využívala aktivní vektory maximálně a nulové nevyužívala vůbec. Na grafu by se to projevilo většími mezerami mezi jednotlivými otočkami.

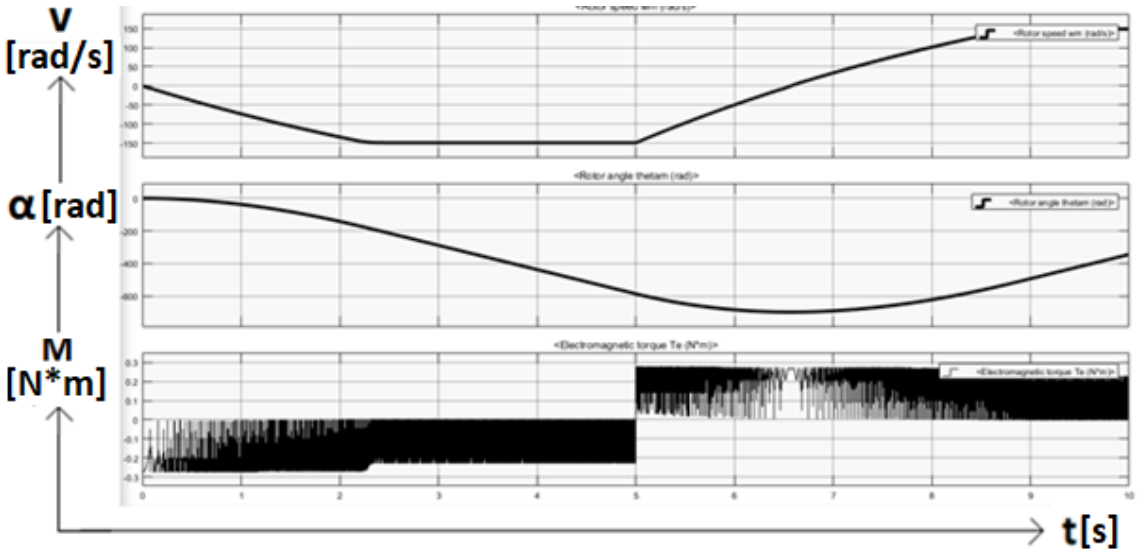
Délka tohoto rotujícího vektoru od středu a úhel natočení představují vstup pro SVM modulaci. Jejich správnost je tedy kritická pro správnou funkčnost.

## 9.2 Pohon do záporných i kladných otáček

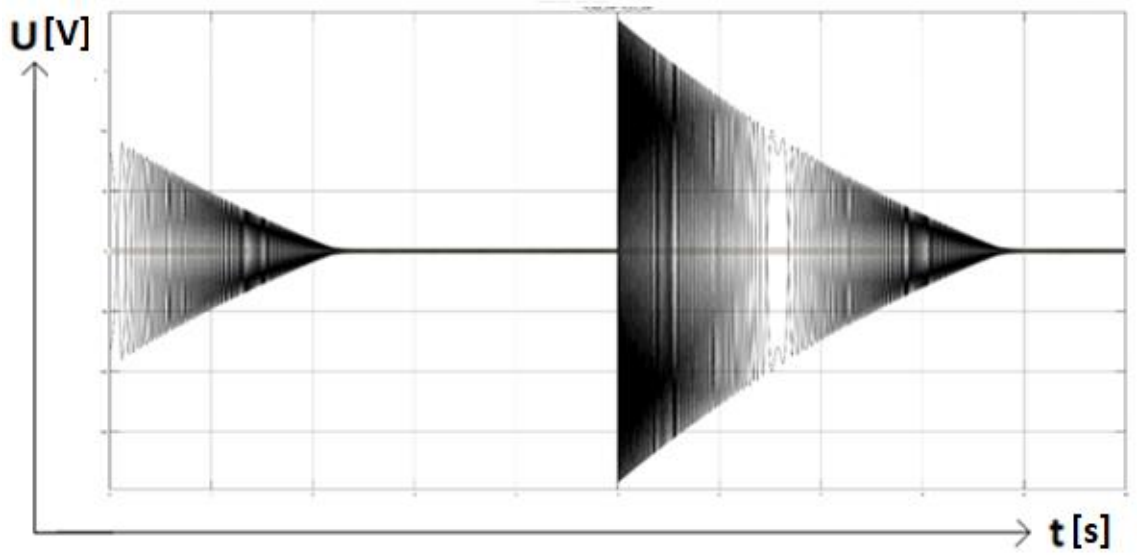
Delší dobu jsem se potýkal s problémem, kdy regulace otáček byla možná pouze zvyšováním jedním směrem. Po důkladném analýze celého řídícího modelu jsem zjistil, že problém nastává při výpočtu délky rotujícího vektoru před vstupem do SVM. Délka vektoru se počítá pomocí Pythagorovy věty a výsledek je vždy kladný. Zapojení jsem tedy doplnil o logiku, která velikosti vektoru přidá záporné

znaménko v případě, že je žádaná rychlost menší jak současná. Řídící model poté reguluje otáčky správným směrem.

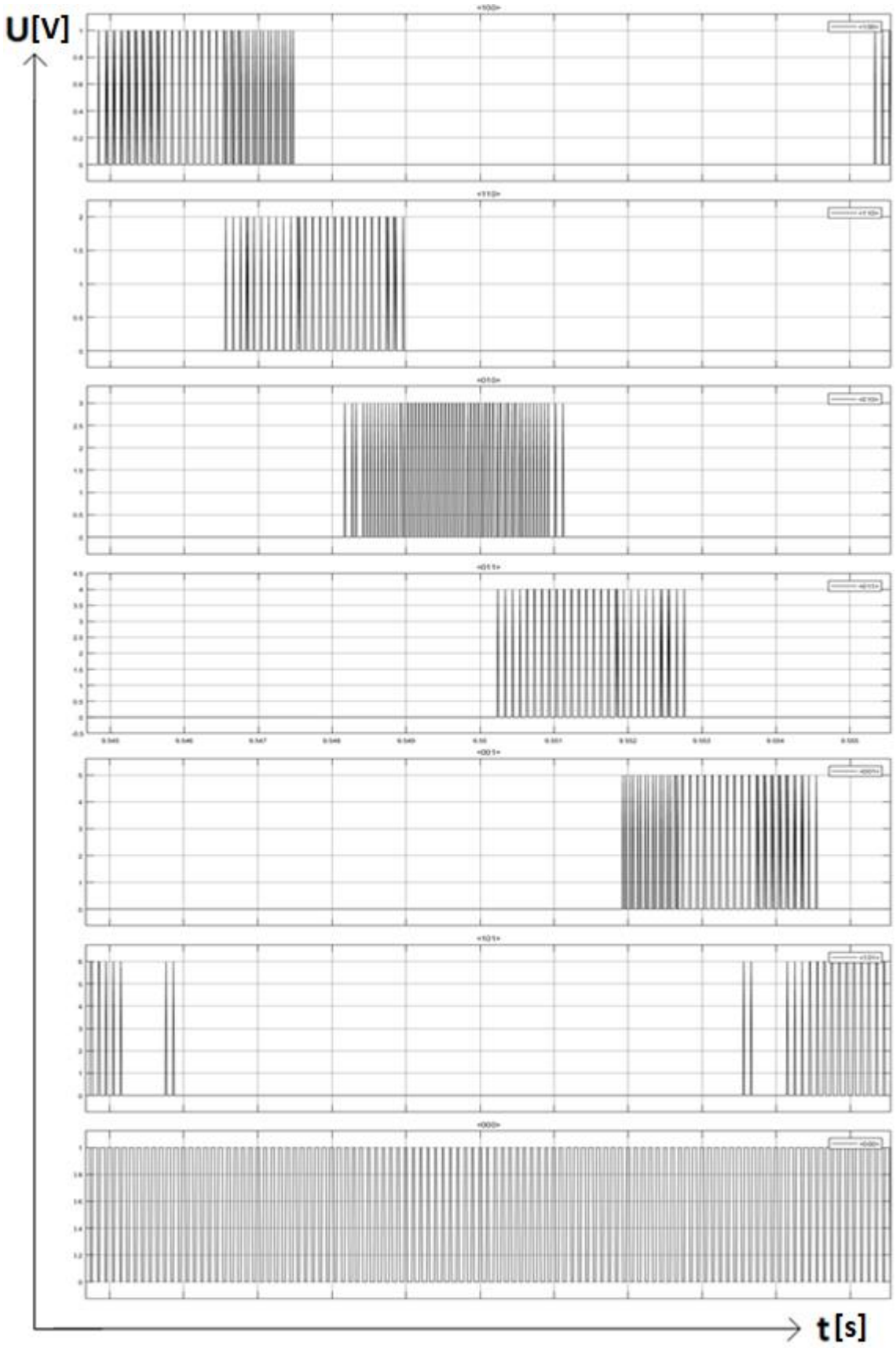
Na Obrázek 24 lze vidět, jak je rychlost regulovaná na hodnotu -100 a v čase t = 5 nastane požadavek na rychlost +100. Můžeme také vidět, že točivý moment změní svou polaritu při regulaci otáček opačným směrem.




**Obrázek 24 Jeden z časových průběhů SVM modulace v závislosti na čase**

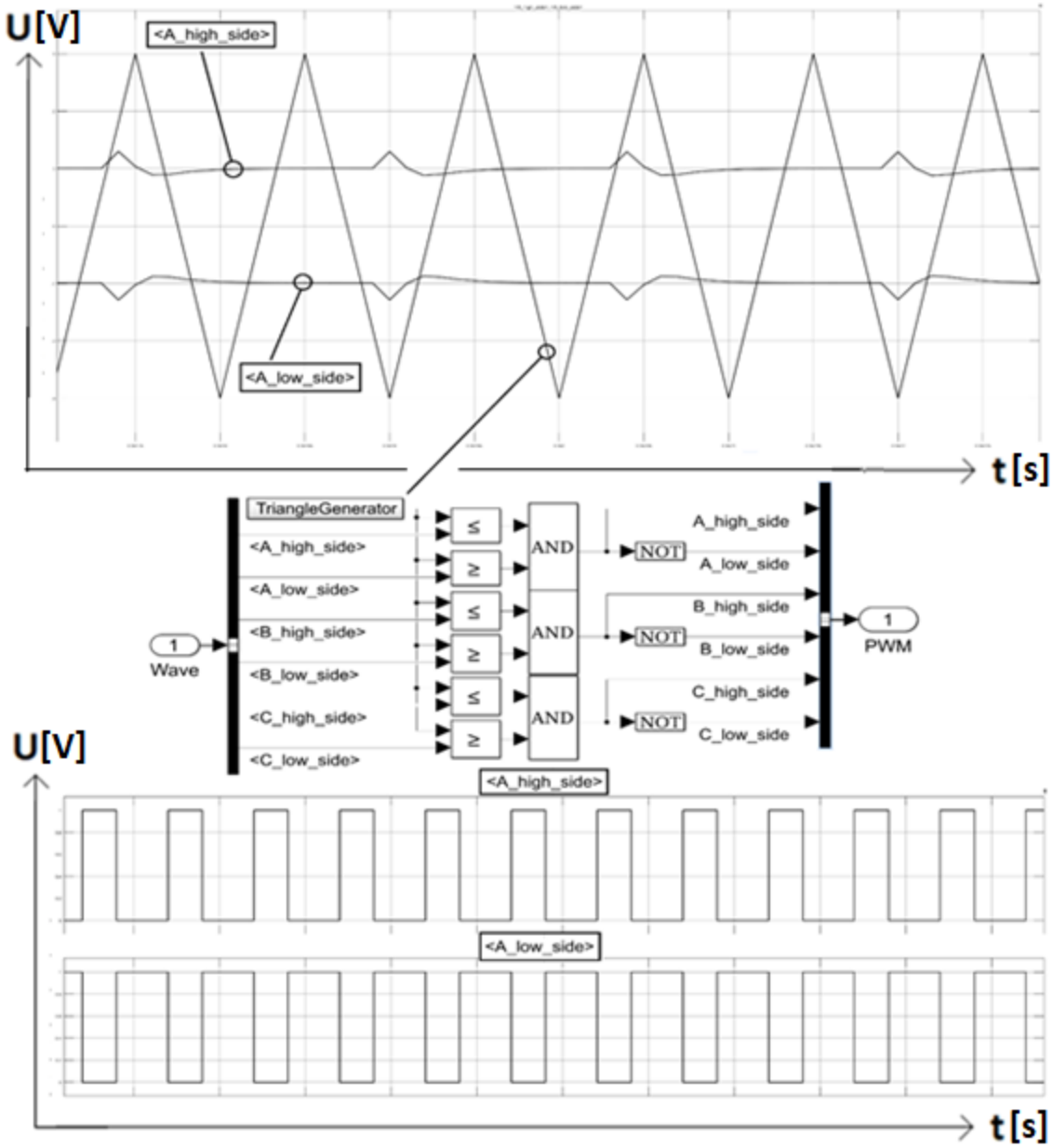



**Obrázek 23 Průběh rychlosti, úhlu a točivého momentu v závislosti na čase**




**Obrázek 25 Využití jednotlivých vektorů polohy v závislosti na čase**

## 9.3 SVM PWM




**Obrázek 26 Průběhy signálů před a po modulaci**

Pro správnost regulace je kritické správné nastavení SVM PWM. Nastavování parametrů jsem prováděl empiricky. Postupoval jsem parametr po parametru a sledoval změny průběhů PWM a otáček motoru.

Na obrázku č. 23 lze vidět, jak se jednotlivých šest stavů mezi sebou prolíná a nikdy nejsou přepínány více než dva aktivní stavy v jedno čase. Pasivní stavy se střídavě přepínají mezi aktivními v závislosti na odchylce od žádané rychlosti. Střída PWM pro každý stav se postupně zvětšuje ke středu a od středu klesá. Kvůli

velkému množství vzorků a rozlišení displeje tento fakt nelze úplně jasně vidět. Na obrázku č. 24 lze pozorovat přiblížený průběh PWM bloku SVM.

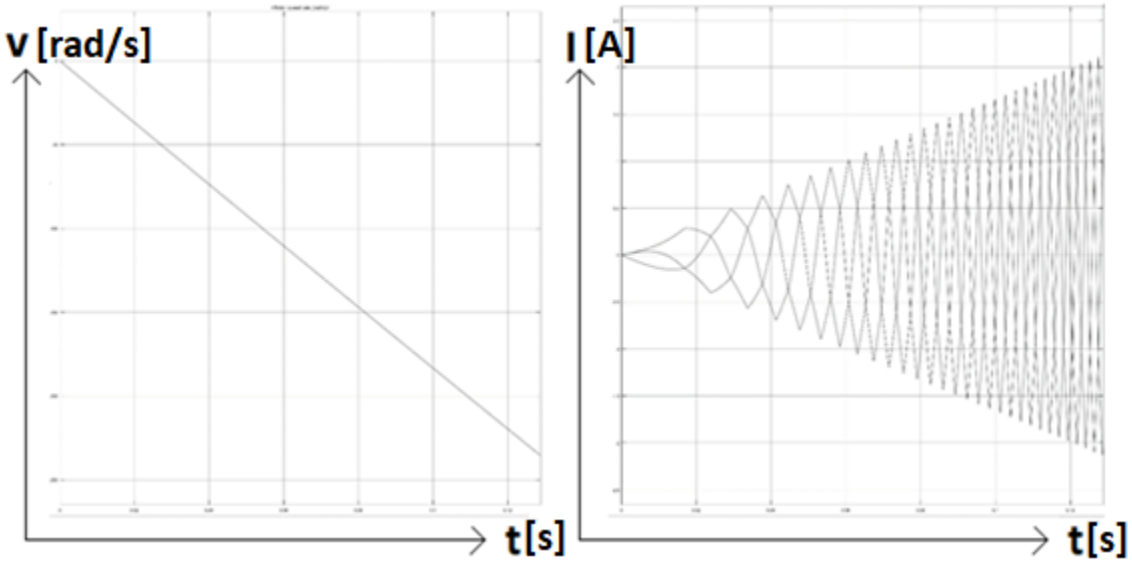
## 9.4 Test zpětného elektromotorického napětí

Při potížích s vytvořením sledovače zpětného elektromotorického napětí jsem potřeboval ověřit, zda napětí, které měřím na přívodních fázích motoru, je skutečně zpětně ovlivněné zpětných elektromotorickým napětím.

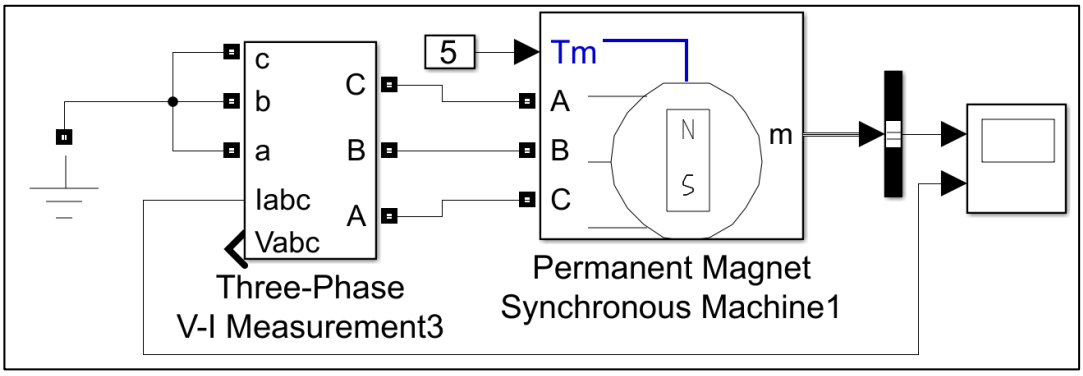
Nastavil jsem modelu motoru konstantní točivý moment a přívody do motoru přes ampérmetr připojil za zem. Ukázalo se, že přívodní fáze motoru fungují skutečně obousměrně.

Na Obrázek 27 můžeme v levé části vidět narůstající rychlost motoru a v pravé části vidíme rostoucí proud jednotlivými fázemi.

Na Obrázek 28 je vidět zapojení použité pro testování.




**Obrázek 27 Rychlost a proud motoru v závislosti na čase**




**Obrázek 28 Zapojení pro testování funkčnosti modelu motoru**

# 10 GENEROVÁNÍ VHDL KÓDU

Simulaci, jež byla vytvořena, lze při splnění určitých podmínek převést do HDL kódu a implementovat do FPGA čipu. FPGA čipy neprovádí výpočty s plovoucí řádovou čárkou za použití FPU, ale v pevné řádové čárce. Tento fakt představoval největší problém, jelikož Simulink ve výchozím nastavení pracuje s plovoucí řádovou čárkou.

## 10.1 Nastavení parametrů bloků modelu pro převod

Pro převod modelu do pevné řádové čárky je zapotřebí každý blok přenastavit do pevný řádové čárky. Dále je zapotřebí určit bitovou šířku signálu a také rozdělit počet bitů pro reprezentaci celých čísel a pro desetinou část.

Určení, kolik bitů je potřeba na vyjádření celých čísel, vychází z maximální předpokládané hodnoty na výstupu daného bloku. Máme-li například šestnáctibitový signál, jehož jeho očekávaná maximální hodnota je 300dec = 0000 000**1 0010 1100**bin, potřebujeme 9 bitů pro vyjádření celých čísel a zbyde nám 7 bitů pro vyjádření desetinné části.

U každého bloku Simulinku lze nastavit jeho předpokládanou maximální a minimální hodnotu. Po stisknutí tlačítka vypočítat nejlepší přesnost Simulink sám určí, kolik je potřeba bitů pro jednotlivé části. Funkci Calculate best precision bohužel nelze volat hromadně a s počtem bloků v modelu je přenastavení zdlouhavé.

Možnost, jak přenastavování urychlit, spočívá v tom, že maximální hodnotu bloků Simulinku lze zadávat jako proměnnou MATLABu. Ve spojitém modelu jsou pomocí bloku TO WORKSPACE zaznamenané hodnoty do prostředí MATLAB.

V Matlabu jsem vytvořil script, který z každé proměnné vyčte maximální a minimální hodnotu a uloží do nových proměnných. Komponenty modelu, který pracuje s pevnou řádovou čárkou, jsou nastavovány za využití hodnot v nových proměnných.

## 10.2 Převod do pevné řádové čárky

Převod původní spojité simulace do pevné řádové čárky je velmi náročný, proto byl vytvořen nový projekt a při každé menší změně bylo ověřeno, zda je HDL kód možné generovat.

Ve spojité simulaci pro přehlednost používám bus creator a bus selector na převod signálů mezi jednotlivými bloky. V projektu, ze kterého se generuje HDL kód, se nepoužívá bus creator ani bus selector, aby ve vygenerovaném HDL kódu byly všechny signály odděleně a bylo možné jednoduché sledování jejich průběhu.

## 10.3 Korektnost diskrétního modelu

Spojitý model, který je považován za referenční, je potřeba porovnávat s nespojitým modelem pro ověření korektnosti převodu a správné funkce.

Porovnání je prováděno vizuální kontrolou použitím bloků scope, avšak pro přesnější porovnání by bylo vhodné posílat hodnoty do prostředí MATLAB a provádět nad každou veličinou výpočet odchylky.

# 11 VÝBĚR GATE DRIVERU

Byl vybrán IO DRV8305, vyráběný přímo za účelem řízení hradel tří fázových měničů. Poskytuje ochranu proti sepnutí obou tranzistorů v jedné větvi, nastavitelný mrtvý čas (zpoždění náběhu do sepnutého stavu a předčasné vypnutí za účelem zamezení otevření obou tranzistorů větve), ochranu proti přehřátí, proti nadproudu, přepětí i podpětí. Chybné stavy indikuje chybový výstup a konkrétní chybný stav může být získán z SPI registru. Poskytuje nastavení velikosti a délky proudu do hradla.

Pro měření proudu IO obsahuje zesilovače, které zesilují úbytek napětí na rezistorech pro měření proudu. Rezistory se umisťují pod dolní tranzistor v každé větvi měniče. Výstupní proud vypočteme ze vztahu:

$$V_0 = \frac{V_{VREF}}{k} - G * (SN_x - SP_x),$$

**Rovnice 10.3-1**

kde VREF je referenční napětí VREG pinu, G je nastavitelné zesílení zesilovače, k = 2 nebo 4 m a SN a SP jsou napětí na vrchní a spodní části rezistoru. Pro nastavení zesilovače potřebujeme znát rozsah výstupního napětí a také maximální proud motoru. Měřené napětí očekáváme kladné i záporné, jelikož spínáme indukční zátěž pomocí PWM a v momentě, kdy je PWM ve vypnuté části své periody neaktivní, teče proud přes jeden rezistor do motoru a přes druhý ven. Pro měření AD převodníkem, jehož rozsah je v intervalu nula až referenční napětí, je potřeba nastavit předpětí, které posune napětí do kladných hodnot.

PWM lze přivádět na gate driver ve více módech:

- Mód 1 – Umožňuje ovládat horní i spodní tranzistor zvlášť
- Mód 2 – Ovládáme pouze horní tranzistor, spodní je vždy v opačném stavu
- Mód 3,4 – Slouží pro 6 krokovou komutaci

# 12 KOMUNIKACE S FPGA

Komunikace podle vzdálenosti můžeme rozdělit následovně:

- Komunikace probíhající v rámci příslušného ESC kontroléru a komunikace mezi FPGA a ESC. Jedná se o komunikace na krátké vzdálenosti do 15 cm
- Komunikace zajišťující vzdálené ovládání – ovládání z rádiového ovladače, či ovládání z chytrých zařízení, např. přes bezdrátové rozhraní Wifi

Hlavní pozornost je věnována komunikacím na krátké vzdálenosti, bez kterých nelze ESC realizovat. Proběhl však také průzkum i testování komunikací na dlouhé vzdálenosti.

## 12.1 Komunikace na krátkou vzdálenost

Při výběru vhodného rozhraní jsem vycházel z vlastního průzkumu nejvíce používaných rozhraní u komponent, které bylo v plánu v bakalářské práci použít. Byl vybrán obvod DRV8305, který využívá rozhraní SPI stejně jako většina AD převodníků, bylo tedy zakomponováno SPI rozhraní stažené od společnosti DigiKey, a to master i slave, protože skutečný hardware ještě nebyl dostupný. SPI není pevně daný standard a vyskytuje se ve více modifikacích, které se liší především velikostí přenášených dat, kdy jsou data někdy přenášena dokonce v konstantním nepřerušovaném streamu, i když nejsou aktuálně dostupná žádná užitečná data k odeslání.

Nejprve došlo k provedení simulací, kterými bylo zjištěno a ověřeno chování stažených komponent. Simulace a poté i zkouška na desce Basys 3 byla provedena za použití tzv. loopbacku, tedy zapojení master i slave komponenty na jeden vodič. Po zprovoznění loopbacku jsem realizoval posílání číselné hodnoty z mikroprocesoru NodeMCU do FPGA (hodnota může například v budoucnu představovat požadovanou rychlost motoru).

## 12.2 Komunikace na dlouhou vzdálenost

Pro komunikaci na delší vzdálenost jsem použil NodeMCU, což je mikroprocesor, který integruje Wi-Fi rozhraní a lze jej tedy použít jako převodník dat typu Wi-Fi do SPI rozhraní a naopak. Programování lze provést Arduino jazykem. Při použití NodeMCU lze využít na straně komunikace na delší vzdálenost dvě rozhraní:

- **SocketServer** – toto rozhraní využívají programy a aplikace
- **WebServer** – jedná se o webovou stránku, kde je pomocí parametrů adresy přenášena numerická hodnota

Za použití návodu na vytvoření webServeru byla vytvořena stránka posílající jako parametr žádosti číselnou hodnotu, kterou NodeMCU přeposílal dál do FPGA

přes SPI rozhraní. Jednalo se spíše o ověření, zda tímto způsobem lze s FPGA komunikovat. Bylo zjištěno, že komunikace je realizovatelná, avšak nevhodná pro komplexní řízení finálního zařízení jako dron, protože vyčítaní hodnot ze žádosti (adresy) není ideální způsob pro přenos více hodnot v jednom čase.

Stejně jako u první varianty byla vytvořena jednoduchá Android aplikace implementující socketServer. Zjistilo se, že se jedná o obtížněji realizovatelnou možnost vyžadující znalost programování Android aplikací v jazyce Java a také znalost soketů. Komunikaci se nepodařilo zprovoznit. V budoucnu však tuto variantu bude možné využít, jelikož nevyžaduje speciální zařízení a stačí pouze mobilní telefon.

## 12.3 Komunikace s DRV8305

Tento driver má registrovou mapu sestávající z jedenácti dvanácti-bitových registrů. První 4 registry jsou indikační (pouze ke čtení), ostatní nastavovací. Indikační registry je důležité číst kvůli ošetření případných chyb a kontrolní registry je vhodné nastavit v závislosti na použitém tranzistoru a typu řízení. Byl implementován systém chybových kódů, který zjednodušuje čtení chyb z registrů.

## 12.4 Komunikace s ADC

Analogově-digitální převodník má pouze jeden kontrolní registr, který ovládá multiplexor a tedy vstup, který je digitalizován. Data lze číst ve dvou módech:

- **Kontinuální** – CS_N (chip-select) udržujeme stále v nule a pouze měníme požadavek na čtenou hodnou (ovládání multiplexoru). Tímto způsobem lze dosáhnout nejvyššího možného toku dat
- **Přerušovaný** – stačí-li nám číst data méně častěji, než 200 kSPS, posíláme požadavek pouze v určitých intervalech

# 13 NÁVRH ALGORITMU PRO POČÁTEČNÍ ROZTOČENÍ MOTORU

Protože implementujeme řízení pro BLDC motor určený pro dron, nemá motor žádný senzor polohy. Při jeho roztáčení je tedy důležité uvést motor do známé polohy, což se provede krátkým proudovým impulsem. Při nastavení vhodných stříd PWM ovládajících jednotlivé fáze je možné přivést motor do jakékoliv polohy. Pro řízení bude použita šesti kroková komutace, a proto je vhodné přivést motor právě do jedné z 6 poloh. Při této variantě můžeme hned první komutací určit směr jeho rotace.

Uvedení do známé polohy je implementováno tak, že jedna fáze je připojená na kladné napětí se střídou v řádu jednotek procent v závislosti na napětí a tvrdosti zdroje. Druhá a třetí fáze je připojena na zem. Nastává zde však problém s vrtulí, která v zarovnané poloze kmitá, což je způsobeno sílou impulzu. Finální verze je tedy inspirována komerčními ESC drivery, které používají techniku postupného zvyšování energie do fáze s následkem odstranění kmitání.

Stejně jako při uvádění motoru do známé polohy, je potřeba implementovat postupný nárust proudu i při roztáčení motoru, při čemž ze známé polohy nemůžeme aplikovat přímo požadovanou rychlost, ale je potřeba v závislosti na zátěži motoru rychlost postupně zvyšovat tak, aby motor stíhal sledovat rotující magnetické pole. Zjištění závislosti rychlosti motoru na čase a závislosti zrychlení na potřebné energii – střídě PWM – při přechodu na výslednou rychlost je klíčová část roztáčení a také následné regulace rychlosti, jak do vyšších, tak do nižších otáček.

# 14 ŘÍDÍCÍ PROGRAM ŠESTIKROKOVÉ KOMUTACE

## 14.1 Čítač s proměnnou rychlostí

Vzhledem k výše uvedeným požadavkům na změnu rychlosti motoru v poměrně velkém rozsahu je potřeba vytvořit děličku hodinového signálu – mezi programátory tzv. povolovací (enable) signál. Klasická implementace děličky hodinového signálu je tvořena registrem. Při určité hodnotě jeho rozsahu (komparační hodnotě) dojde ke změně logické úrovně sníženého hodinového signálu. Rychlost motoru nastavujeme jako hodnotu v intervalu od nuly po maximální rychlost. U klasické děličky popsané výše by v případě použití hodnoty rychlosti motoru jako komparační hodnoty s vyšší hodnotou byla frekvence pulzů nižší. Výsledek by tedy byl opačného charakteru, než se očekává. Komparační hodnota by se musela invertovat na určitém intervalu nula až maximální rychlost motoru. Tohoto lze docílit jednodušeji jinou implementací.

Princip děličky, vhodné pro náš případ, spočívá oproti klasické děličce v tom, že místo navyšování čítače v každém hodinovém taktu o jednotku, navyšujeme čítač o vstupní hodnotu (rychlost), která může být větší nebo rovna jedné. Čím vyšší je tato navyšovací (increase) hodnota, tím rychleji čítač přeteče, což způsobí rychlejší změny nového výsledného hodinového signálu.

Vzorce pro tuto komponentu:

- Pro výpočet increase hodnoty pro libovolnou frekvenci použijeme vzorec:
  - inc_value = (required pulse rate / (100*(10^6)) ) * 2^number_of_bits
- Nejnižší frekvenci pro určitou délku registru spočítáme pomocí vzorce:
  - bottom frequency = (100*(10^6)) / 2^number_of_bits
- Frekvenci při znalosti increase hodnoty a velikosti čítače spočítáme:
  - any frequency = inc_value * (100*(10^6)) / 2^number_of_bits
- Maximální hodnotu frekvence, které lze dosáhnout s touto komponentou:
  - maximal frequency = 50*(10^6)

## 14.2 Struktura hardwarového popisu ve VHDL

Struktura hardwarového popisu ve VHDL se liší od struktury softwarových programů. Hardwarový popis se skládá z hierarchicky na sebe propojených komponent. Je zvykem označit hlavní komponentu slovem top. Top komponenta je jediná komponenta, která může přistupovat k pinům FPGA. Dále se signály propagují do dalších komponent a vzniká hierarchický design. Hardwarový popis

je vhodné rozdělovat do sub-komponent podle funkce, aby bylo možné snadno znovu využít určitou část kódu bez potřeby úpravy – v nejlepší, případě pouze vytvoření instance a napojení vstupů a výstupů bez úprav.

### 14.2.1 Komponenta top_VHDL

Jedná se o hierarchicky nejvyšší komponentu programu, jež je použita pouze pro vytvoření instancí dalších komponent – neobsahuje logické funkce. Je zároveň jedinou komponentou, jejíž porty lze mapovat na výstupní a vstupní piny.

### 14.2.2 Komponenta communication

Tato komponenta slouží pro oddělení komunikačních komponent od zbytku programu. V této práci byla nakonec použita pouze SPI master komponenta, ale v budoucnu budou použity i další již otestované a funkční komponenty: spi_slave_logic, i2c_master_logic, transmitter_logic, reviever_logic.

Lze si všimnout, že komponenta communication vytváří instance komponent končících **_logic**. Nejedná se totiž o poslední komponenty v hierarchii této větvě komunikačních komponent. **_logic** byly nazvány, protože implementují stavové automaty řídící odesílání dat do stejnojmenných komponent, které již v názvu neobsahují **_logic**.

Stejnojmenné komponenty neobsahující **_logic** implementují samotný algoritmus daného komunikačního protokolu. Jelikož se jedná o sériové linky, je tedy obvykle na jedné straně vždy vektor dat pro odeslání s řídícími signály pro zahájení transakce a indikace dokončení transakce, a na druhé straně serializovaná data, vhodná pro přenos po sériových linkách. Tyto komponenty se, na rozdíl od těch implementující logiku, musí zabývat časováním daného protokolu a přidáním arbitrážních signálů jako parita.

Obecně se tedy jedná o zpřehlednění a oddělení logiky od samotného algoritmu rozhraní pro serializaci dat na výstupní piny.

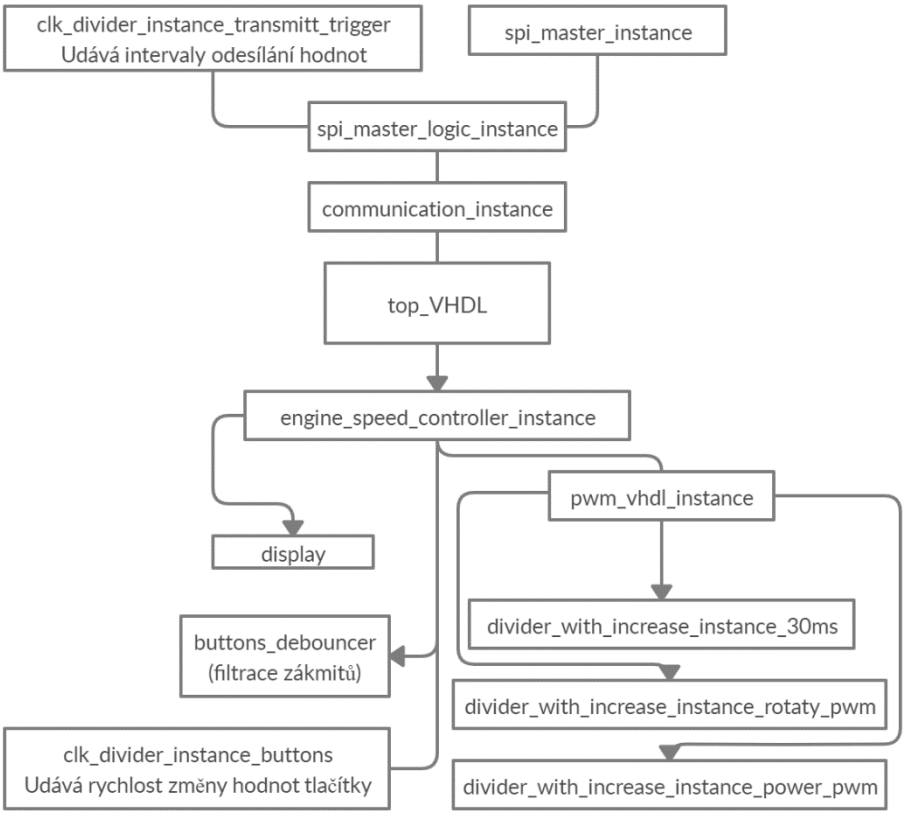
### 14.2.3 Komponenta engine_speed_controller

Jedná se o hlavní řídící komponentu pro řízení motoru. Tato komponenta obsahuje stavový automat, jež určuje fázi řízení. Bylo považováno za vhodné vložit do této komponenty také komponentu ovládání displeje, protože jsem ho v dané době využíval primárně k zobrazování střídy PWM a rychlosti motoru. S postupem času se začal používat i pro zobrazení dat z SPI a stálo by za to přehodnotit nejvhodnější umístění v hierarchii. Komponenta také obsahuje debouncer (komponenta odstraňující zákmity při stisku tlačítek) pro 5 tlačítek, jež se nachází na desce Basys 3. Pro konkrétní motor je potřeba nastavit tyto parametry:

- Rychlost náběhu střídy a její mezní hodnoty při uvádění motoru do známé polohy
- Rychlost změny periody cyklu při roztáčení motoru
- Změna střídy PWM za jeden cyklus při roztáčení motoru

Tyto parametry je těžké vyladit pro konkrétní motor, a proto je struktura stavového automatu přizpůsobena tak, aby ladění parametrů bylo co nejsnazší. Přechody mezi jednotlivými stavy jsou prováděny stisknutím tlačítka. V každém stavu lze nastavit hodnotu, a to buď PWM střídy, nebo rychlosti motoru pro další stav. Takto lze snadno pozorovat, zda konkrétní stav řízení pracuje správně. Na displeji je vždy zobrazovaná aktuálně nastavovaná hodnota pro další stav.

Stavový automat v této komponentě se stará primárně o přechody mezi stavy v závislosti na stisknutí tlačítka a o nastavování hodnot. Tato komponenta vytváří instanci komponenty **pwm_vhdl**, jejíž činnost je plně závislá na stavech stavového automatu v této komponentě. **pwm_vhdl** však stavy nemůže měnit, pouze se jimi řídí.

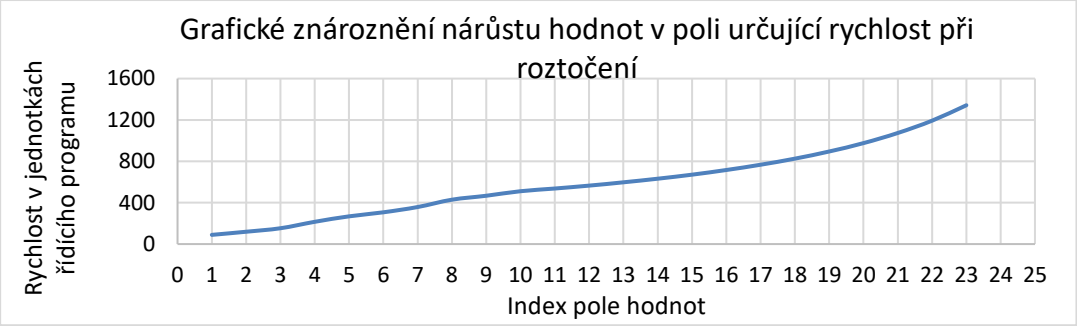



**Obrázek 29 Blokový diagram hierarchie komponent hardwarového popisu**

## 14.3 Popis stavů stavového automatu

### 14.3.1 RESET

Tento stav je výchozí stav po naprogramování FPGA a zároveň do něj lze přejít kdykoliv stisknutím prostředního tlačítka desky Basys3. Na displeji se neukazuje nic, je vypnutý. Stav neposkytuje nastavení parametrů pro další stav. PWM výstupy pro motor jsou na hodnotě 0, motorem tedy neteče žádný proud. Stav PREPOSITION PHASE. V tomto stavu dojde k uvedení motoru do známé polohy metodou popsanou výše, při které aktivujeme určitou fázi motoru a předpokládáme posun motoru na danou pozici.




**Obrázek 30 Grafické znázorněné nárůstu hodnot v poli pro roztočení motoru**

### 14.3.2 CHANGE OF ROTATY PWM IN TIME – pole

Jedná se o první stav, kdy se motor začíná točit. V tomto stavu využíváme pole hodnot, které bylo vyladěno pro motor, jež je použit v této bakalářské práci. Hodnoty představují rychlosti, které se z pole postupně čtou s každou další komutací – protože hodnoty udávají rychlost, s každou další hodnotou čteme v menším intervalu než předchozí.

### 14.3.3 CHANGE OF ROTATY PWM IN TIME – konstanta

Tento stav je schopný motor přivést na finální požadovanou rychlost. Rychlost, jež není získávána z pole hodnot, ale navyšována o konstantní hodnotu. Bylo ovšem potřeba implementovat logiku, která podle rychlosti náběhu a aktuální rychlosti určí hodnotu střídy PWM – rychlý náběh na určitou rychlost vyžaduje větší energii než při pomalém náběhu na stejnou rychlost.

### 14.3.4 ROTATY PWM SPEED

V tomto stavu se motor již točí a další stavy slouží pouze pro nastavování hodnot. V komponentě **pwm_vhdl**, která slouží pro oddělení nastavování hodnot od

samotného řízení, je řízení definováno jako výchozí stav stavového automatu, aby **společné stavy pro nastavování hodnot** dále neovlivňovaly samotné řízení.

### 14.3.5 ROTATY PWM SPEED

Umožňuje měnit rychlost motoru.

### 14.3.6 POWER DUTY

Tento stav umožňuje měnit požadovanou střídu při ladění programu. Finální stav programu cílí k automatickému nastavování střídy.

### 14.3.7 POWER PWM SPEED

Tento stav umožňuje měnit frekvenci PWM. Nicméně frekvence byla stanovena na 20kHz a není ji potřeba měnit.

### 14.3.8 ADC VALUES

Umožňuje listovat tlačítky nahoru a dolů hodnotami aktuálně měřenými na AD převodníku. Hodnoty se mění velice rychle, proto bylo přidáno umělé zpomalení obnovovací frekvence zobrazování hodnot. Jedná se spíše o ověření, zda AD převodník měří, než o využitelné rozhraní pro čtení hodnot. V tomto stavu lze také vyčíst chybové kódy z driveru:

- OV – overvoltage – přepětí, OC – overcurrent – nadproud,
- UV – undervoltage – podpětí, OT – overtemperature – přehřátí

**Tabulka 3 Přiřazení chybový kódů parametrům informačních registrů**

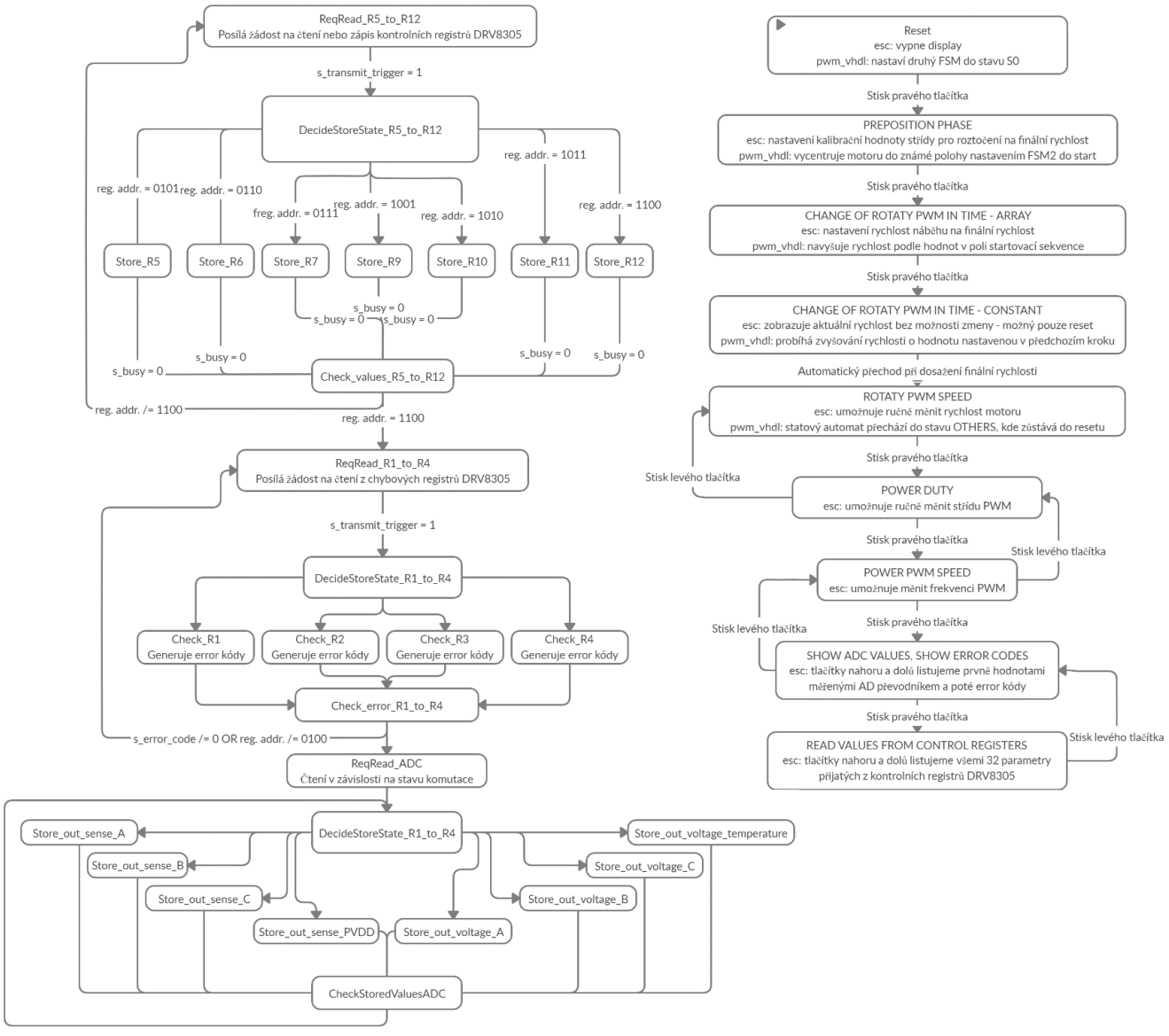
| Číslo | Popis chyby | Číslo | Popis chyby | Číslo | Popis chyby |
|---|---|---|---|---|---|
| 1 | CONT. REG. WRITE ERR | 13 | OC LS MOSFET A | 25 | AVDD UV FAULT |
| 2 | FAULT INDICATION | 14 | OC HS MOSFET B | 26 | LS GATE SUPPLY FAULT |
| 3 | TEMP FLAG 175 °C | 15 | OC LS MOSFET B | 27 | HS CAHRGE PUMP UV 2 FAULT |
| 4 | PVDD UNDERVOLTAGE | 16 | OC HS MOSFET C | 28 | HS CHARGE PUMP OV FAULT |
| 5 | PVDD OVERVOLTAGE | 17 | OC LS MOSFET C | 29 | HS CHARGE PUMP ov ABS FAULT |
| 6 | VDS status | 18 | SENSE C OC FAULT | 30 | HS MOSFET A DRIVE FAULT |
| 7 | CHARGE PUMP UV FLAG | 19 | SENSE B OC FAULT | 31 | LS MOSFET A DRIVE FAULT |
| 8 | TEMP FLAG 105 °C | 20 | SENSE A OC FAULT | 32 | HS MOSFET B DRIVE FAULT |
| 9 | TEMP FLAG 125 °C | 21 | PVDD UV 2 FAULT | 33 | LS MOSFET B DRIVE FAULT |
| 10 | TEMP FLAG 135 °C | 22 | WATCHDOG FAULT | 34 | HS MOSFET C DRIVE FAULT |
| 11 | OT WARNING | 23 | OT FAULT | 35 | LS MOSFET C DRIVE FAULT |
| 12 | OC HS MOSFET A | 24 | VREG UV FAULT | -- | --- |

## 14.3.9 CONTROL REGISTER OF DRV8305

Umožňuje čtení hodnot jednotlivých parametrů kontrolních registrů driveru DRV8305. Č. – číslo, H. – hodnota, R. – registr.

**Tabulka 4 Předpokládané hodnoty pro při čtení kontrolních registrů**

| Č. | Název | H. | R. | Č. | Název | H. | R. | Č. | Název | H. | R. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | IDRIVEP_HS | 4 | 5 | 12 | SET_VCPH_UV | 0 | 9 | 23 | GAIN_CS3 | 00 | 10 |
| 2 | IDEIVEN_HS | 4 | 5 | 13 | CLR_FLTS | 0 | 9 | 24 | CS_BLANK | 00 | 10 |
| 3 | TDRIVEN | 11 | 5 | 14 | SLEEP | 0 | 9 | 25 | DC_CAL_CH1 | 0 | 10 |
| 4 | IDRIVEP_LS | 4 | 6 | 15 | WD_EN | 0 | 9 | 26 | DC_CAL_CH2 | 0 | 10 |
| 5 | IDRIVEN_LS | 4 | 6 | 16 | DIS_SNS_OCP | 0 | 9 | 27 | DC_CAL_CH3 | 0 | 10 |
| 6 | TDRIVEP | 11 | 6 | 17 | WD_DLY | 01 | 9 | 28 | VREV_UV_LVL | 10 | 11 |
| 7 | TVDS | 10 | 7 | 18 | EN_SNS_CLAMP | 0 | 9 | 29 | DIS_VREG_PWRGD | 0 | 11 |
| 8 | TBLANK | 01 | 7 | 19 | DIS_GDRV_FAULT | 0 | 9 | 30 | SLEEP_DLY | 01 | 11 |
| 9 | DEAD_TIME | 1 | 7 | 20 | DIS_PVDD_UVLO2 | 01 | 9 | 31 | VREF_SCALE | 01 | 11 |
| 10 | PWM_MODE | 00 | 7 | 21 | GAIN_CS1 | 00 | 10 | 32 | VDS_MODE | 0 | 12 |
| 11 | COMM_OPTION | 1 | 7 | 22 | GAIN_CS2 | 00 | 10 | 33 | VDS_LEVEL | 25 | 12 |




**Obrázek 31 Vývojový diagram pro FSM SPI a ESC komponenty**

## 14.4 Komponenta pvm_vhdl

Obsahuje stavový automat, který se řídí stavy z komponenty **engine_speed_controller**. Obsahuje druhý stavový automat, jehož stav je nastavován ve stavech prvního stavového automatu. Druhý stavový automat se stará čistě o vytváření PWM a o komutaci. Tato komponenta využívá komponentu **divider_with_increase**, jejíž princip je vysvětlený v kapitole 14.1 pro nastavení rychlostí všech tří používaných čítačů. Účel každého čítače:

**divider_with_increase_instance_30ms – čítač** je využit v PREPOSITION PHASE stavu a jeho rychlost ovlivňuje délku periody, během které se zvyšuje střída motoru – viz. výše postupné ustálení motoru do počáteční polohy bez kmitání.

**divider_with_increase_instance_rotaty_pwm** – generuje frekvenci v závislosti na nastavené hodnotě rychlosti pomocí tlačítek.

**divider_with_increase_instance_power_pwm – generuje** frekvenci v závislosti na nastavené hodnotě střídy pomocí tlačítek.

### 14.4.1 Popis druhého FSM zajišťující komutaci motoru

V této bakalářské práce byla použita šestikroková komutace. Při níž jsou vždy dvě fáze aktivní a třetí je tzv. plovoucí, protože není připojená ani na VDDA, ani na GND. Je v tzv. stavu vysoké impedance, což je v ideálním případě stav, kdy se chová, jako by nebyla do obvodu vůbec zapojena.

Zpětné elektromotorické napětí je přímým důsledkem Faradayova zákona, který zjednodušeně říká, že při pohybu vodiče v magnetickém poli je ve vodiči indukováno napětí [39]. Fáze tedy v momentě, kdy je připojená na stav vysoké impedance, míjí permanentní magnet statoru a je do ní indukováno napětí. Toto napětí však není jednoduché snímat při použití unipolární PWM (na rozdíl od bipolární PWM, která dosahuje hodnoty v intervalu +VDDA do – VDDA, obsahuje unipolární hodnoty v intervalu +VDDA do 0 V), protože díky zapojení motoru do hvězdice na fázi naměříme součet onoho indukovaného napětí (Faradayův zákon) a napětí přicházejícího z uzlu spojujícího všechny 3 fáze do hvězdice. Toto napětí je v tomto případě PWM, ovládající zbylé dvě fáze.

Abychom naměřili pouze indukované napětí, je potřeba snímat v momentě, kdy je PWM, která řídí ostatní dvě fáze ve vypnutém stavu. Z tohoto předpokladu můžeme odvodit dva požadavky:

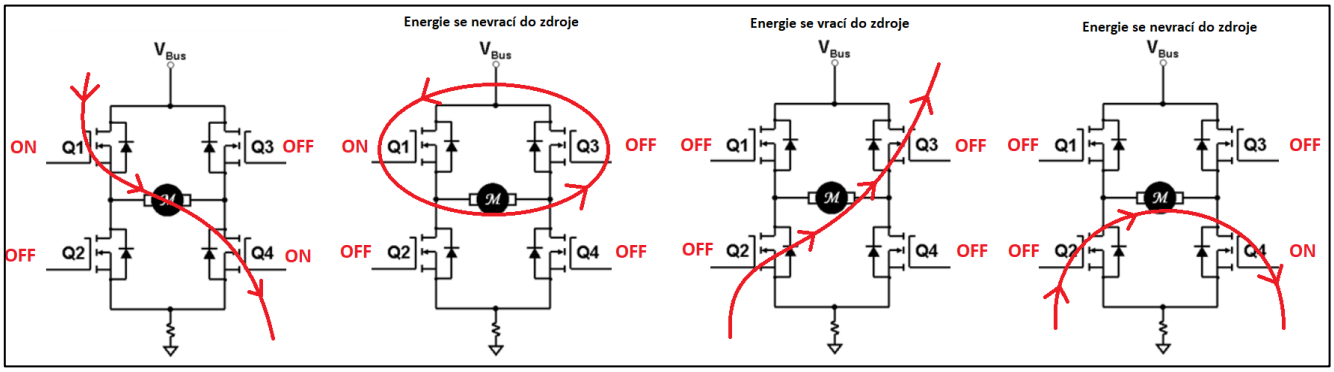
- PWM nesmí dosáhnout 100% střídy – snímáme, když je vypnutá
- Frekvence a fáze AD převodníku musí být synchronizovaná s PWM driveru

Dalším důležitým požadavkem je spínat pouze horní MOSFETy měniče a spodní používat pouze pro komutaci fází. Kdybychom spínali pouze spodní MOSFETy, z uzlu, který spojuje všechny 3 fáze, by nepřicházelo vhodné napětí

PWM, nýbrž pevné napětí VDDA. Indukované napětí bychom přes napětí VDDA nebyli schopní měřit. V případě spínání horních i spodních tranzistorů bychom viděli dvě různé PWM, což není vhodné.

Indukované napětí je rovněž možné detekovat během zapnutého stavu PWM. V tomto případě by průchod nulou (zero crossing) nastal v momentě průchodu polovinou napětí VDDA [31].

### 14.4.2 Možné druhy spínání můstku [33]




**Obrázek 32 Možné druhy spínání měniče [33]**

Pro zjednodušení byly možné stavy při vypnutém stavu PWM znázorněny na můstku pro DC motor – toto zjednodušení lze provést, jelikož je použita šestikroková komutace, kdy je jedna větev vždy ve stavu vysoké impedance.

Na prvním můstku zleva vidíme tranzistory Q1 a Q4 otevřené – proud teče motorem. Další tři můstky demonstrují tok proudu při různých způsobech spínání.

**2. můstek zleva** – byl uzavřen tranzistor Q4 a energie naakumulovaná v cívce se vybíjí na parazitní diodě tranzistoru Q3 a přechodu drain-source tranzistoru Q1.

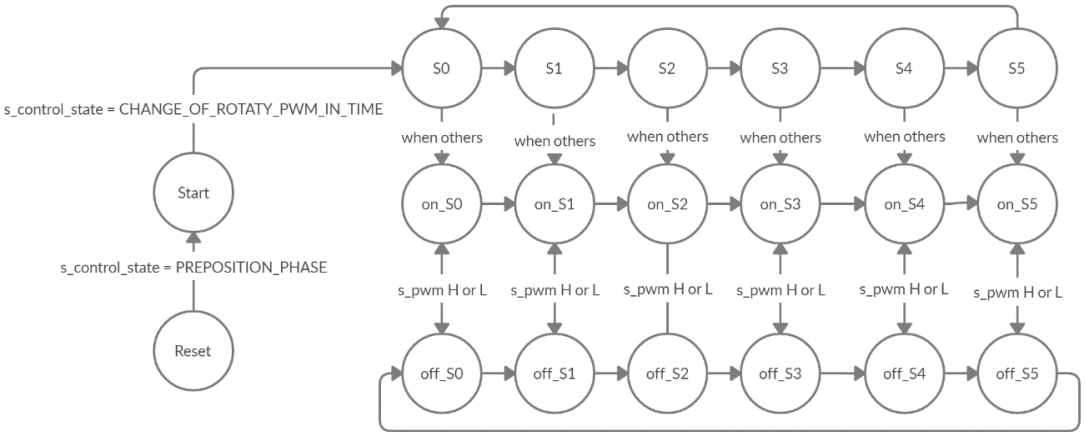
**3. můstek zleva** – byl uzavřen tranzistor Q1 i Q4. Napětí naindukované na motoru posílá energii přes parazitní diodu Q3 a Q2 do zdroje.

**4. můstek zleva** – byl uzavřen tranzistor Q1 a energie naakumulovaná v cívce se vybíjí na parazitní diodě tranzistoru Q2 a přechodu drain-source tranzistoru Q4.

Pro detekci zpětného elektromotorického napětí potřebujeme spínat pouze horní tranzistory. Proud poteče ve vypnutém stavu stejně jako na 4. můstku z leva. Stavový automat jsem rozdělil do dvaceti stavů:

- **Reset** – všechny výstupy vypnuté
- **Start** – slouží k uvedení motoru do pevné polohy – 3 z 6 spínačů aktivované
- **S0, S1, S2, S3, S4, S5** – použito při prvních 23 komutacích, kdy je rychlost čtená z pole hodnot. PWM se přivádí na horní i spodní tranzistory
- **on_S0, on_S1, on_S2, on_S3, on_S4, on_S5** – tyto stavy jsou aktivní, když je PWM ve stavu HIGH. Horní i spodní tranzistor příslušných větví je otevřený

- **off_S0, off_S1, off_S2, off_S3, off_S4, off_S5** – tyto stavy jsou aktivní, pokud je PWM ve stavu LOW. Horní tranzistor přepne příslušné větvě do vypnutého stavu a aktivuje spodní tranzistor stejné větve – dojde k vyzkratování cívky




**Obrázek 33 Vývojový diagram pro FSM komutace motoru**

## 14.5 Komponenta spi_master_logic

Implementuje stavový automat, který nejprve komunikuje s DRV8305 a poté s ADC.

### 14.5.1 Stavy pracující s kontrolními registry driveru

- **ReqRead_R5_to_R12** – posílá požadavek na zápis do registru driveru 5 až 12
- **DecideStoreState_R5_to_R12** – Stav implementuje potřebné zpoždění 1 takt
- Store_R5, Store_R6, Store_R7, Store_R9, Store_R10, Store_R11, Store_R12
  - Stavy pro zpracování hodnot – parsování a uložení
- **Check_values_R5_to_R12** – Zajišťuje cyklické čtení a posun do další fáze

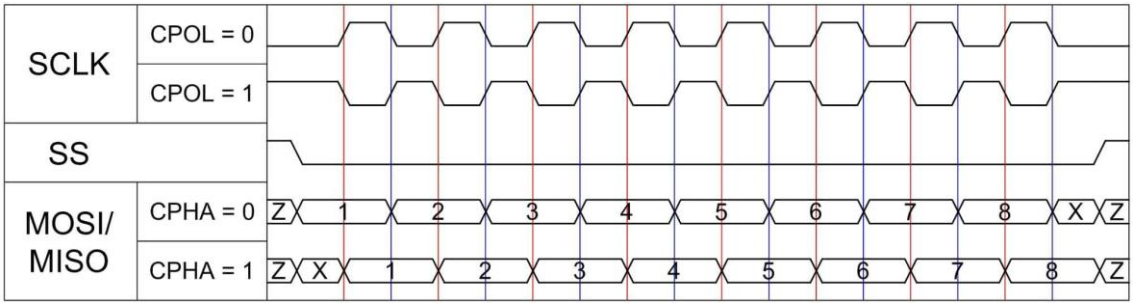
### 14.5.2 Stavy pracující s informačními registry driveru

- **ReqRead_R1_to_R4** – posílá požadavek na čtení z registru driveru 1 až 4
- **DecideStoreState_R1_to_R4 R12** – Stav implementuje zpoždění o 1 takt
- Check_R1, Check_R2, Check_R3, Check_R4 generuje chybové kódy
- **Check_error_R1_to_R4** – při chybě nepokračuje dál a cyklicky čte registr

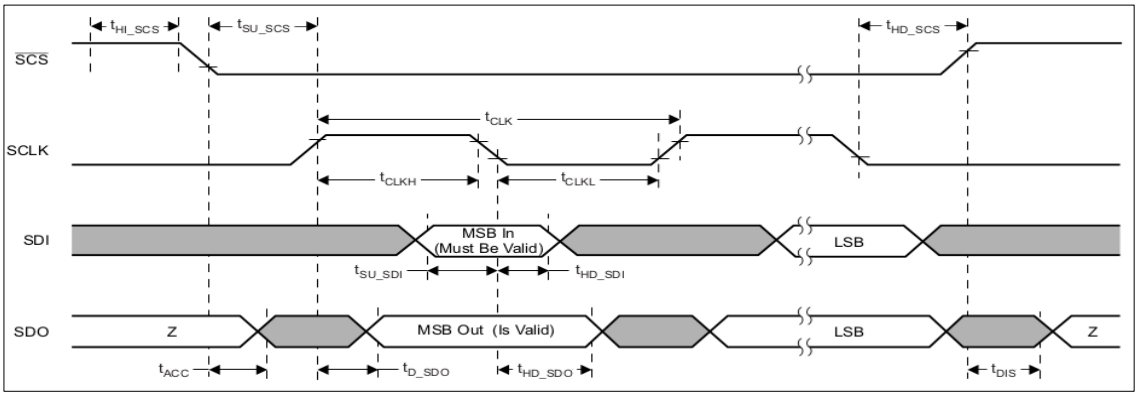
### 14.5.3 Stavy komunikují s ADC

- **ReqRead_ADC** – posílá požadavek na konverzi konkrétního vstupu ADC
- **DecideStore_ADC** – Stav implementuje potřebné zpoždění 1 takt

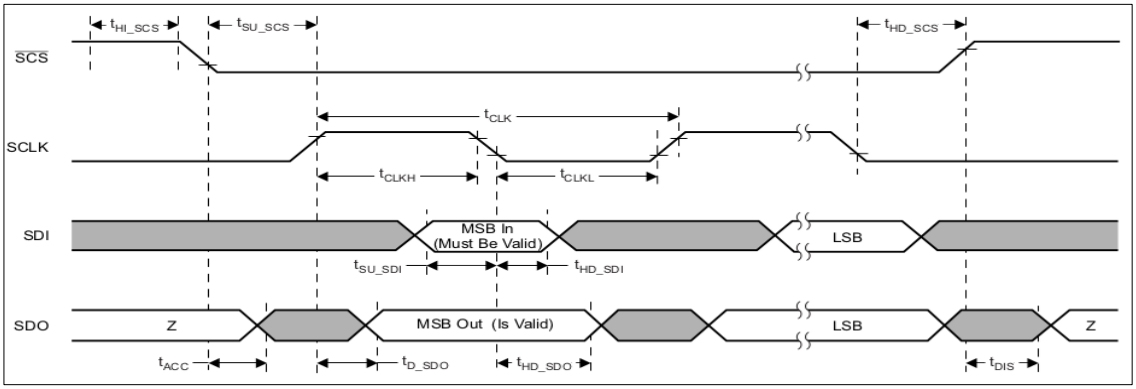
- Store_out_sense_A, Store_out_sense_B, Store_out_sense_C, Store_out_sense_PVDD, Store_out_voltage_A, Store_out_voltage_B, Store_out_voltage_C, Store_out_temperature, CheckStoredValuesADC
  Stavy ukládající hodnoty pro použití v komponentě pwm_vhdl a display




**Obrázek 36 Pomůcka pro nastavení polarity a fáze SPI rozhraní [103]**




**Obrázek 35 Časový diagram pro SPI rozhraní DRV8305 [3]**



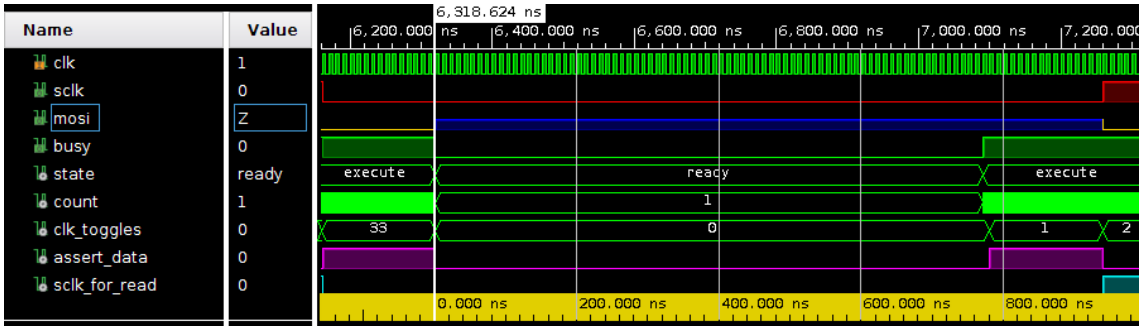

**Obrázek 34 Časový digram pro SPI rozhraní analogově digitálního převodníku [45]**

## 14.5.4 Zjištění nastavení pro SPI DRV8305 a ADC

Podle Obrázek 36 můžeme rozhodnout o potřebném nastavení polarity a fáze SPI signálu. Na Obrázek 35 z technické specifikace DRV8305 [32] vidíme, že hodinový signál je v nečinném stavu v hodnotě nula, jeho polarita je tedy podle Obrázek 36

nulová a fáze hodinového signálu je jedna, protože jsou data posílány na sestupné hraně.

Na Obrázek 34 z technické specifikace ADC108S022 [32] lze vidět, že hodinový signál je jedna v klidovém stavu. Polarita je tedy jedna. Fáze hodinového signálu je jedna, protože při výchozí polaritě hodinového signálu (což je polarita nula), by byla data odesílána na sestupnou hranu. Vzhledem k tomu, že řídící deska byla navržena tak, že DRV8305 i ADC108S022 sdílí stejné SPI rozhraní, je potřeba před každou transakcí nastavit tyto parametry znovu.




**Obrázek 37 Záznam ze simulace ukazující 500ns prodlevu mezi transakcemi**

Specifikace udává požadavek, aby slave select byl alespoň 500ns ponechán ve stavu logické úrovně 1 – slave-select je aktivní pouze ve stavu execute

### 14.5.5 Simulace funkčnosti odesílání dat z spi_master komponenty


**Obrázek 38 Záznam ze simulace transakce s DRV8305**

Ověření pro **DRV8305**, posíláme data pro 5 registr '0' – write bit, "0101" – adresa pátého registru, “01101000100" – data


**Obrázek 39 Záznam ze simulace transakce s ADC**

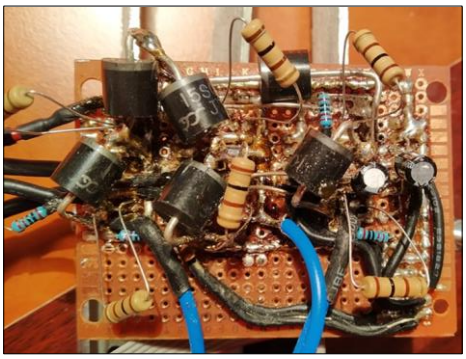
Ověření pro **ADC108S022** posíláme požadavek na hodnotu AD vstupu číslo 5 (101), odesílané slovo je 11**101**1111111111. Přijatá data simulací ověřit nelze, protože bychom museli disponovat zdrojovým kódem pro ADC.

# 15 NÁVRH PCB S VYUŽITÍM IR2110

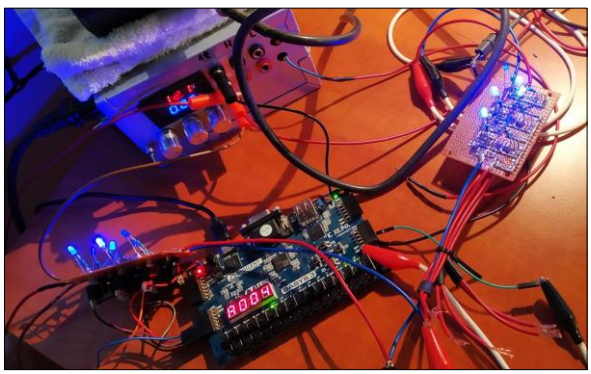
První plošný spoj byl navržen v prostředí Altium designer. Byla to moje první zkušenost s tvorbou plošného spoje a zároveň prostředí pro pokročilé uživatele. Hlavní překážkou byl výběr schématických značek a foot-printů pro dané součástky. Kromě toho, importování stažených komponent není příliš jednoduché. Foot-printy byly staženy ze stránky https://www.ultralibrarian.com/. Jednalo se o návrh zapojení, které bylo realizováno na nepájivém poli (viz. níže) pro ověření, zda je zapojení správně. Tento plošný spoj nebyl dotažený do konce, jelikož zapojení na nepájivém poli potvrdilo nevýhody použití obvodu IR2110. V zapojení chybí gate rezistory a kapacity pro napájení můstku. Schéma v příloze.

# 16 REALIZACE DRIVERU S IR2110

Tento driver byl realizován pro vyzkoušení algoritmů pro řízení na reálném hardwaru. Zapojení bylo realizováno na nepájivém poli a přineslo četnou řadu nevýhod. Cesty tvořené pájkou jsou náchylné na vytvoření zkratu. Obvod driveru nepatří k nejjednodušším obvodům, a bylo tedy potřeba použít drátové propojky, které přispívají k nepřehlednosti zapojení. Byl použit gate driver IR2110, který dodá až 2 ampéry proudu do hradla tranzistoru. Zapojení mi pomohlo částečně odladit řídící algoritmus, avšak nebylo ale s velkou pravděpodobností správné, protože se spálilo velké množství obvodů. Celkově byly vyrobeny 4 verze zapojení. Na video záznamu je vidět, že bez zátěže se motor zvládne roztočit. Tento gate driver neintegruje tzv. dead-time (uměle přidaná doba mezi sepnutím horního a vrchního tranzistoru větve), což mohlo vést k vysokým proudům. Příčinou poruch byl také použitý zdroj, jelikož nemá omezení proudu. Realizace tohoto HW byla velmi naučná a často vyžadovala hledání chyb. Od zapojení bylo ve správnou dobu upuštěno a pokračovalo se návrhem plošného spoje s více pokročilým driverem od Texas Instruments DRV8305, který umožňuje nastavení dead-time a má řadu ochranných obvodů.



**Obrázek 40 Spodní část zapojení driveru na pájivém poli**



**Obrázek 42 Testovací sestava s IR2110**

# 17 NÁVRH SCHÉMA ZAPOJENÍ DRIVERU S DRV8305 [3][43]

Po předchozích zkušenostech s návrhem plošného spoje v Altium Designer bylo potřeba zvolit jiný návrhový program, který bude jednodušší a méně náročný. Po důkladném průzkumu dostupných nástrojů jsem zvolil KiCad, což je open-source multiplatformní EDA software. Aktuální verze KiCadu neobsahuje autorouter a je potřeba použít externí openSource autorouter s názvem freeRouting.jar.

## 17.1 Rozvržení schématu

Schéma jsem rozdělil na tzv. hierarchické listy. Hierarchii lze tvořit více způsoby:

- **Kaskádovitě** (do hloubky) – struktura podobná objektovému programování
- **Plochá struktura** – hlavní list obsahuje všechny ostatní listy

Zvolil jsem plochou strukturu, protože návrh není rozsáhlý. Při návrhu schématu v KiCad je možné realizovat propojení mezi jednotlivými listy pomocí:

- **Globálních proměnný** – propojení nejsou vidět na první pohled
- **IN/OUT výstupů listů** – propojení formou viditelných čar

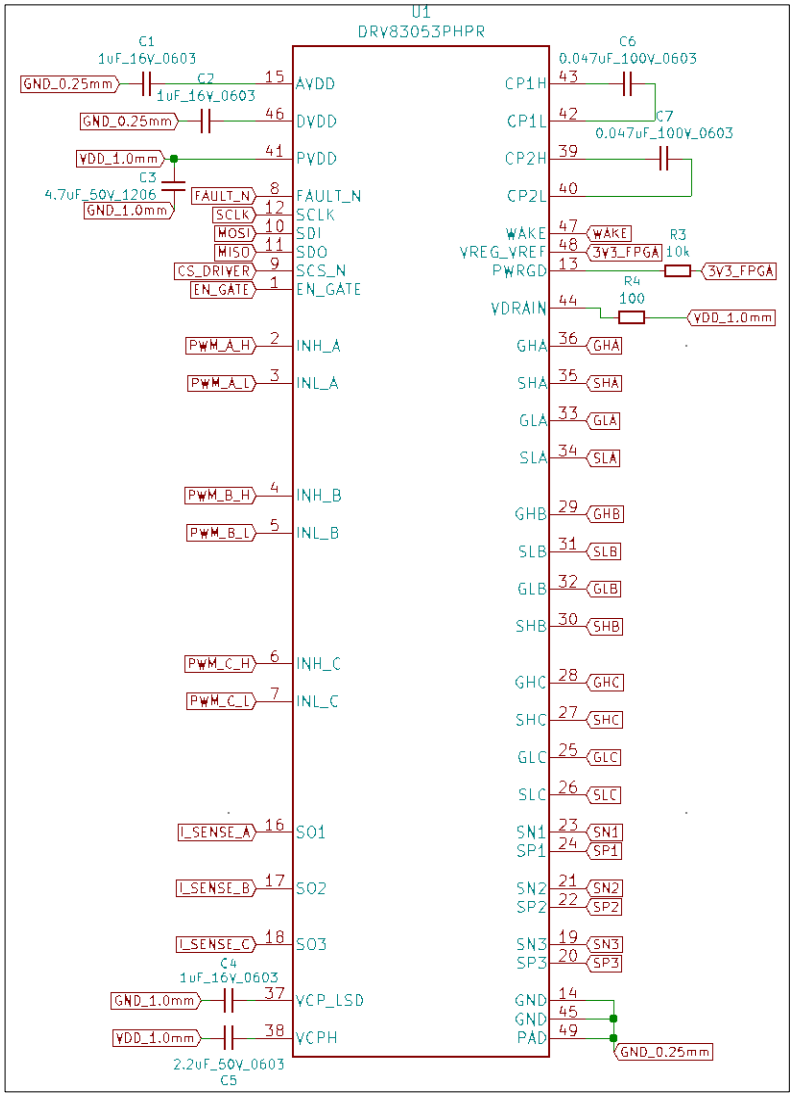
Zvolil jsem propojení pomocí globálních proměnných, protože design je „čistější“
Návrh driveru je rozdělen do 4 listů:

- **Driver** – zde je realizováno zapojení gate driveru DRV8305
- **Bridge** – zde je realizováno zapojení řídícího můstku – měniče
- **ADC** – zde je realizováno zapojení analogově-digitálního převodníku
- **IO** – rozvody digitálních i analogových signálů a jejich připojení na piny

## 17.2 Výhody použití DRV8305 oproti IR2110

DRV8305, na rozdíl od IR2110, umožňuje pomocí SPI rozhraní nastavit dead-time. Zapojení s DRV8305 bude možné využít i pro vektorové řízení, které jsem realizoval v semestrální práci jako simulaci v Simulinku díky možnosti měření proudů větvemi můstku. DRV8305 obsahuje ochranu proti přehřátí, kterou IR2110 nemá a je tedy velmi jednoduché obvod spálit. DRV8305 je patrně levnější varianta a integruje všechny drivery v jednom pouzdře, na rozdíl od IR2110, u kterého je potřeba použít zvlášť obvod pro každou větev řídícího můstku.

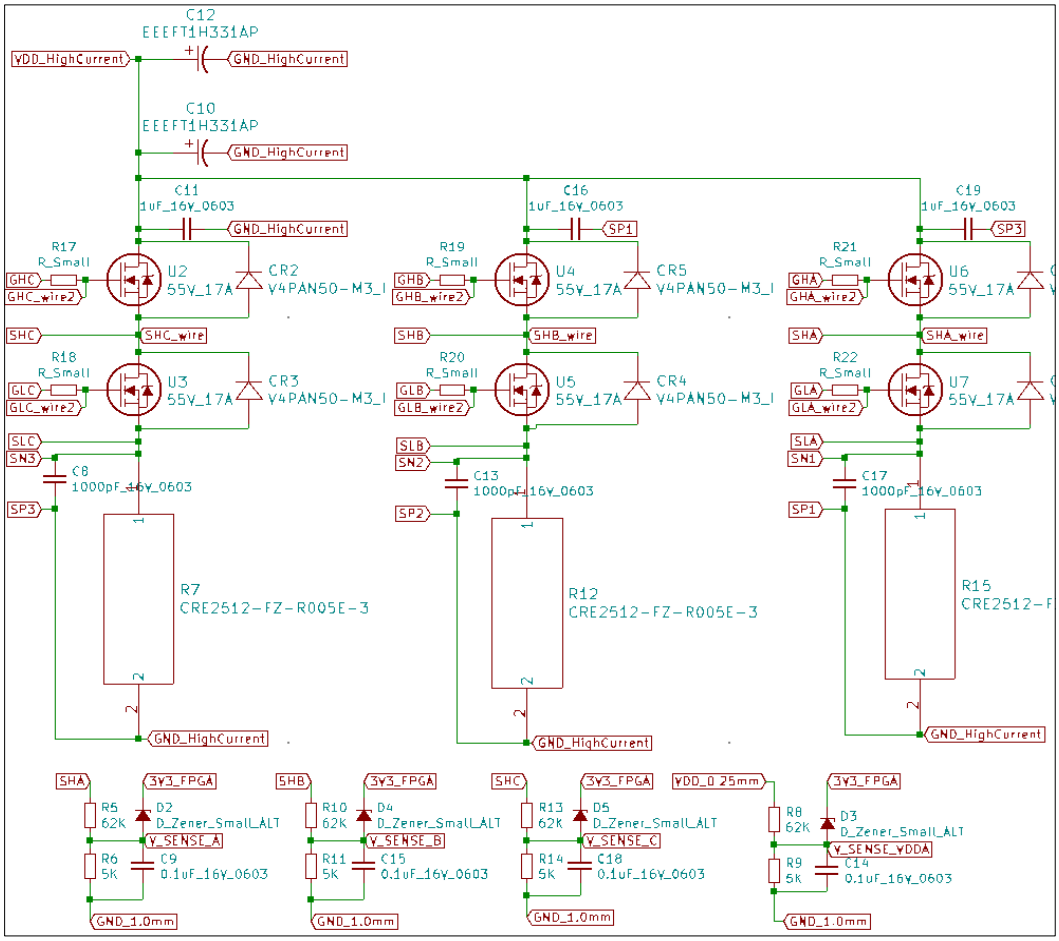
## 17.3 Zapojení DRV8305 [3]




**Obrázek 43 Zapojení DRV8305**

Zapojení je celé dimenzované pro napětí VDDA = 50 V. Kondenzátory **C6**, **C7** je dle specifikace potřeba dimenzovat na dvojnásobek VDDA, proto byly zvoleny kondenzátory na 100 V. Vstup PWRGD informuje mikročip o správnosti připojených napětí. Pro správnou funkčnost je potřeba provést připojení přes 10kΩ rezistor připojený na napájení. VDRAIN je vstup, který informuje o velikosti napětí VDDA – napájecí napětí můstku – napětí na drainu horních tranzistorů. Kondenzátory **C1** a **C2** jsou externí prvky vnitřního analogového a digitálního regulátoru napětí IC. PVDD je napájecí vstup mikročipu. Kondenzátor **C3** na vstupu je největší jak rozměrově, tak kapacitou na plošném spoji. Má kapacitu 4.7µF a je dimenzovaný na 50 V. Kondenzátory **C4** a **C5** jsou externí prvky vnitřního low-side a high-side gate budiče. Kondenzátor **C5** stačí dimenzovat na 16 V, díky nízkému rozdílu ceny oproti 50 V jsem zvolil variantu s rezervou.

## 17.4 Zapojení můstku




**Obrázek 44 Zapojení měniče**

Na Obrázek 44 vidíme zapojení měniče. Na přívodu napájení jsou připojené dva 330µF filtrační kondenzátory. V zapojení na každém drainu vrchních MOSFETů je připojený 1µF kondenzátor, který by měl být na plošném spoji umístěný co nejblíže samotnému pinu gatu. Tento kondenzátor je potřeba, protože samotný tranzistor může mít tendenci oscilovat a kondenzátor oscilacím zamezí. Na gate vstupu každého MOSFETU je připojený tzv. gate rezistor, který umožňuje rychlejší vybití parazitních kapacit při vypnutí, omezuje proud do hradla a zamezuje oscilacím. Pájecí plošky pro připojení motoru jsou připojené na SHA_wire, SHB_wire, SHC_wire. To je tedy mezi horním a spodním MOSFET tranzistorem v každé větvi měniče. Pod spodním MOSFETem je zapojený rezistor pro měření proudu, který má v tomto návrhu konkrétně hodnotu 0.007 Ω. Měření napětí provádí AD převodník (viz. zapojení níže). Předtím je však napětí pomocí vodičů SNx a SPx přivedeno do DRV8305, kde je zesíleno pomocí operačních zesilovačů. Dělič napětí slouží ke snížení napětí, protože napájecí napětí by poškodilo AD převodník. Dělič má na spodním rezistoru kondenzátor a tvoří tedy dolnofrekvenční propust. Pro nízké kmitočty se chová jako obyčejný napěťový

dělič. Pro vysoké kmitočty se chová jako zkrat. Mezní frekvenci můžeme vypočítat dle vztahu [38]:

$$\omega_0 = \frac{R_1+R_2}{R_1*R_2*C} = \frac{62000+17400}{62000*17400*0.0000001} = 736\frac{rad}{s}$$

**Rovnice 17.4-2**

$$f_0 = \frac{\omega_0}{2\pi} = \frac{736}{2*3{,}14} = 117\ Hz$$

**Rovnice 17.4-1**

Shottkyho dioda, která je ve schématu chybně označena jako zenerova dioda, je připojena na referenční napětí AD převodníku. Referenční napětí je napětí, které vstupní signál nesmí přesáhnout. Dioda by v případě vyššího potenciálu na vstupu AD převodníku svedla napětí zpátky do zdroje. Volíme Shottkyho diodu, protože je velmi rychlá a má malé propustné napětí.

Vzhledem k tomu, že 4 článková Li-Pol baterie má při plném nabití 14.8 V a potřebujeme napětí snížit na 3.3 V, byly zvoleny hodnoty pro dělič napětí 62 kΩ a 17.4 kΩ, které napětí sníží na:

$$U_2 = U_{BAT} * (R2 / (R1 + R2)) = 14.8 * (17400 / (62000 + 17400)) = 3.24V$$

**Rovnice 17.4-3**

Musíme počítat s určitou tolerancí vzhledem k toleranci odporů děliče. Dělič snižuje hodnotu napětí na:

$$100/\ UBAT * U2 = 100/14.8 * 3.24 = 21\ \%\ původního\ napětí$$

**Rovnice 17.4-4**

Při normalizaci v FPGA budeme hodnotu v rozsahu 0 až 3.3 V násobit hodnotou 4.56. Bude-li tedy na převodníku výstupní hodnota 1001101011(bin) = 619(dec), spočítáme hodnotu měřeného napětí jako:

$$U_{výst} = 3.3V\ /\ 1024\ *\ 619\ =\ 1.99V,$$

**Rovnice 17.4-5**

což je hodnota napětí na výstupu děliče. Skutečné měřené napětí získáme jako:

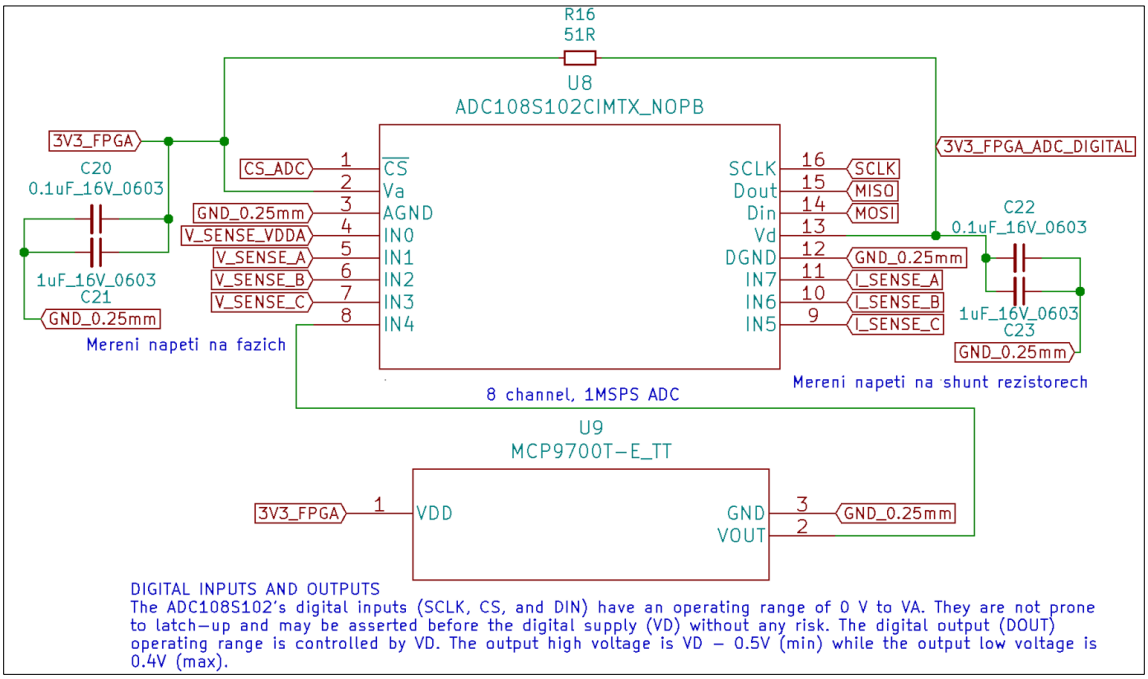
$$U_{skutečné} = U_{výst} *\ 4.56 = 1.99V\ *\ 4.56\ =\ 9.0744V$$

**Rovnice 17.4-6**

Tímto způsobem bylo postupováno při návrhu. Po následné realizaci obvodu je v tomto místě naměřeno napětí maximálně 600mV. To je dané tím, že je používán tvrdý počítačový ATX zdroj a střídu není potřeba nastavovat ani přes 10

% pro nejvyšší otáčky. Při použití jiného zdroje může nastat potřeba pro zvýšení střídy a měřené napětí by tedy bylo vyšší.

## 17.5 Zapojení analogově-digitálního převodníku [45]




**Obrázek 14.5.5 Zapojení ADC**

### 17.5.1 Napájení

AD převodník potřebuje zvlášť napájení pro **analogovou** a **digitální** část. Napájení digitální části musí mít menší napětí než napájení analogové části. V zapojení lze vidět, že 3.3 V je prvně připojeno do **Va** vstupu a přes 51 Ω rezistor dále připojeno na vstup **Vd.** Tato realizace zajišťuje splnění výše uvedené podmínky. Analogová i digitální část má připojený 1µF pro filtrování vstupního napětí a 0.1µF pro zamezení vysokofrekvenčním oscilacím obvodu.

### 17.5.2 Analogové vstupy

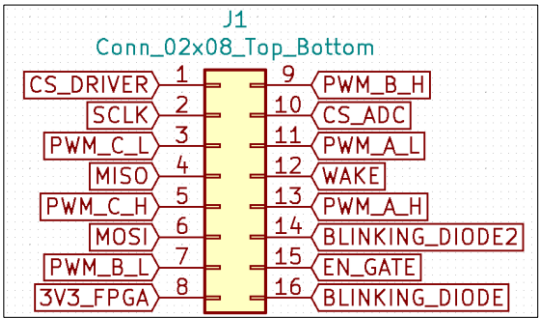
AD převodník má 8 vstupů, z toho **3 jsou využity pro měření napětí na fázích** (měří se výstup děliče napětí). Další 3 vstupy jsou využity pro měření výstupního napětí operačních zesilovačů mikročipu DRV8305, které zesilují napětí měřené na rezistorech pro měření proudu. Jeden vstup je využit pro měření napětí VDDA – také sníženo děličem. Poslední vstup AD převodníku je využit pro **měření teploty**. Teploměr je vidět na schématu pod AD převodníkem (**U9**). Jedná se o jednoduchý integrovaný teploměr, jehož výstup je v rozmezí nula až po hodnotu napájení. Je tedy napájen 3.3 V, aby nepřekročil vstupní napětí AD převodníku.

### 17.5.3 Digitální výstup – MISO

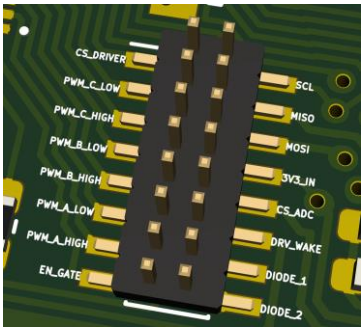
Převedené hodnoty analogových veličin jsou posílány do FPGA přes SPI rozhraní pracující na frekvenci 3.2 MHz.

### 17.5.4 Digitální vstup – MOSI

Do AD převodníku se posílá pouze požadavek na čtení konkrétního vstupu.




**Obrázek 47 Zapojení pinů konektoru**




**Obrázek 46 3D model pinů**

## 17.6 Zapojení pin headeru

Pořadí jednotlivých vstupů a výstupů bylo voleno s ohledem na snadné zapojení na PCB. FPGA umožňuje (vyjma 3V3_FPGA pinu) nastavit pořadí pinů programově.

## 17.7 Zapojení lineárního napěťového regulátoru

FPGA potřebuje 5 V a 1 A napájení. V případě, že drivery budeme napájet z 4 článkové Li-pol baterie, která má 14.8 V, bude potřeba toto napětí pro FPGA snížit. Potřebujeme-li vytvořit určité napětí a opomeneme napěťové děliče a zenerovy diody, máme dvě možnosti, a to buď použít LDO (lineární napěťový regulátor), nebo tzv. buck konvertor.

LDO vyžadují pouze externí kondenzátor, jsou však méně účinné, protože snížení napětí provádí převodem na teplo. Obsahují tepelnou pojistku – je-li výstup regulátoru zkratovaný, může se zdát, že LDO nefunguje, ale příčina může být pouze aktivovaná tepelná pojistka.

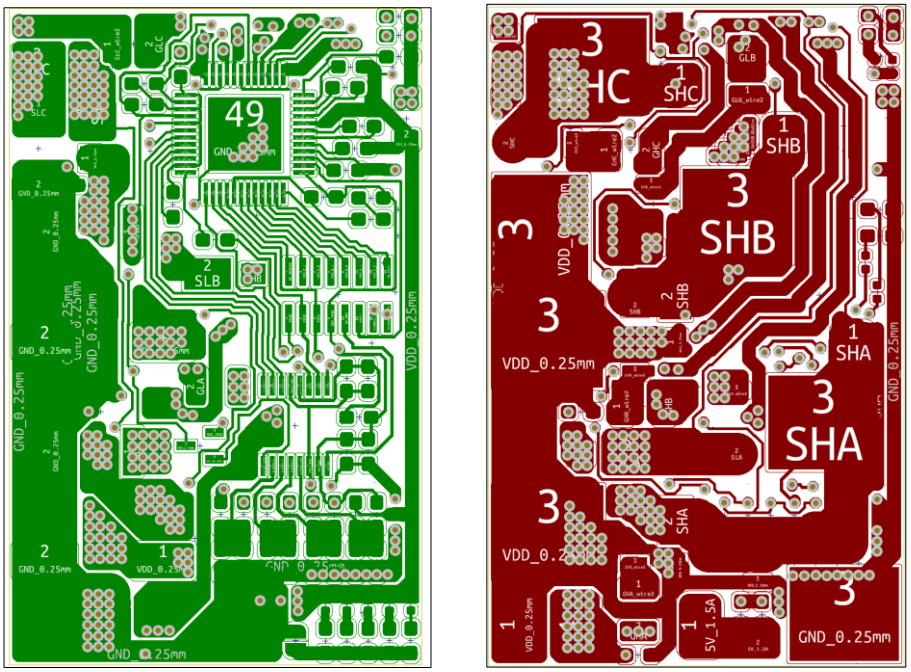
Buck konvertor umožňuje snížit vstupní napětí. Vyžaduje externí induktor, ale poskytuje větší účinnost.

Byl zvolen LDO pro jednoduché zapojení, a s ohledem na proud odebíraný do FPGA nebudou ztráty příliš velké. Na výstupu LDO byl připojen 1µF kondenzátor pro stabilizaci výstupního napětí a 0.1µF k zamezení vlastních oscilací regulátoru. Kondenzátor na vstupu není potřeba, jelikož rozvod napájecího napětí je připojen na dostatek kondenzátorů.

# 18 NÁVRH PLOŠNÉHO SPOJE DRIVERU S DRV8305 [36]

Při návrhu plošného spoje je důležitá základní znalost názvů vrstev používaných jak při návrhu, tak i těch vyskytujících se na finálním výrobku. Každá vrstva vyjma vrstvy Edge.Cuts (obrys PCB) má minimálně přední a zadní stranu. Nejdůležitější vrstvy:

**CU vrstva** – tato vrstva představuje fyzické měděné spoje mezi komponenty
**Mask** – představuje obvykle zelenou krycí vrstvu plošného spoje
**Silk** – vrstva představující popisky na vyrobeném plošném spoji
**CrtYd** – z angl. country yard – představuje ohraničení prostoru součástek




Při návrhu plošného spoje je důležité správně dimenzovat šířky vodičů. Potřebnou šířku lze vypočítat na základě znalosti šířky použité mědi, tolerance ohřevu vodičů, a především požadavku na proud. Výpočet byl realizován pomocí online nástroje. U tohoto plošného spoje bylo použito čtyř různých šířek vodičů. Proud je závislý na požadavku na ohřev, proto hodnotu označuji jako přibližnou.

- 5 mm (cca 10 A) – rozvod VDDA a GNDA pro řídící můstek
- 3 mm (cca 6 A) – propojení horního a spodního MOSFETu větve můstku
- 1 mm (cca 2 A) – propojení gate driveru a gate vstupu tranzistorů
- 0.2 mm (500 mA) – signálové cesty – např. měření napětí, SPI komunikace

Šířka cesty není jediná věc, kterou požadovaný proud ovlivňuje. Musíme také vhodně dimenzovat VIAs, což jsou prokovy (spoje) mezi jednotlivými vrstvami plošného spoje. Propojky mají dva parametry – šířku pro vrtání a celkovou šířku

pro vyrobení propojky včetně okruží. Šířka vrtu byla zvolena 0.4 mm a celková šířka byla zvolena 0.8 mm, z čehož lze odvodit šířku okruží - 0.4 mm. Proud propojky těchto rozměrů je přibližně 2 A na jednu propojku. Dimenzování propojek tedy spočívalo ve zvýšení jejich počtu. Pro 0.2 mm širokou cestu stačí jedna propojka. Pro cesty široké 1 mm byly použity minimálně dvě a pro cesty široké 3 mm a 5 mm minimálně 5 propojek. Kde to bylo možné, bylo propojek vloženo více. Při vkládání propojek na cesty širší, než je cesta, není problém, avšak u 0.2 mm cest je vhodné vložit do místa propojky měděnou plochu o průměru 0.8 mm – velikost propojky. Je tím zajištěno kvalitnější propojení. Velikost plošného spoje byla navrhována podle velikosti křídla konstrukce dronu. Byla zvolena velikost 30 mm x 50 mm.

## 18.1 Rozdělení vrstev v závislosti na proudech:

**Horní vrstva** – především pro nízké proudy – DRV8305 s externími kondenzátory a rezistory, ADC, teploměr, indikační LED diody a piny. **Spodní vrstva** – výkonová část – MOSFET tranzistory a regulátor napětí

## 18.2 Volba umístění elektrolytických kondenzátorů

Obvyklé umístění na komerčních výrobcích je mimo driver na napájecí piny. Není to však pravidlo. Byly zvoleny SMD kondenzátory o kapacitě 2x 330 μF.

## 18.3 Autorouter vs. ručního propojení

Pro nejoptimálnější umístění komponent na plošný spoj je potřeba rozmístění co nejvíce optimalizovat a v některých případech je použití autorouteru nadále zbytečné. Znalost designéra umožnuje lepší propojení. Součástky byly propojeny manuálně.

## 18.4 Chlazení DRV8305 [3]

Byla vybrána varianta čipu s pouzdrem HTQFP-48, který má po celé spodní straně čipu chladící plošku. U vícevrstvých desek je možnost použít jinou vrstvu, než chladič, nicméně u této dvouvrstvé desky je nejlepší udělat propojení na druhou stranu plošného spoje a zde vložit pájecí plošku pro připojení externího chladiče.

## 18.5 Dimenzování MOSFET tranzistorů

Byly zvoleny 17 A MOSFETy poskytující rezervu. Volit nižší proud nemá smysl, protože cena dále neklesá a volit vyšší proud není dobré, protože tranzistory mají větší QG – kapacita gatu a spínání by spotřebovalo více energie. $Q_G$ = 8 nC.

## 18.6 Volba externí diody

Byla zvolena 8 A dioda, jelikož diody na vyšší proud se vyrábí pouze v pouzdrech DPAK a větších. Použití dalších šesti DPAK by znemožnilo splnit rozměry PCB. Přestože se parazitní dioda tranzistoru běžně používá při řízení motorů, použití externí diody má své opodstatnění [37]. Externí dioda má úbytek napětí v propustném směru pouze 570mV, což je 2,5x nižší úbytek parazitní diody. Platí, že ztrátový výkon přeměněný na teplo je rovný součinu úbytku napětí a proudu. Externí dioda má tedy 2.5x menší tepelné ztráty. Další výhodou je, že tepelné ztráty jsou odváděny pouzdrem diody, namísto pouzdrem tranzistoru.

## 18.7 Kondenzátory

Velikost kondenzátorů se odvíjí od poměru požadované hodnoty a maximálního napětí. Byla zvolena velikost 0603 (2x menší než 1206) – tyto kondenzátory jsou obvykle do 16 V. Byla také použita velikost 1206 – VDDA – 4.7 µF 50 V.

## 18.8 Rezistory [40][41][42]

Velikost vybíráme podle ztrátového výkonu. Většina rezistorů má velikosti 0603. Velikost předřadných odporů diod je 0402. Rezistory pro měření proudu větvemi můstku byly zvoleny o velikosti 2512 (přestože R = 7mΩ) kvůli vysokému proudu.

## 18.9 Volba teploměru

DRV8305 má vlastní teploměr pro případný SHUTDOWN, a protože měření teploty je spíše informační, stačilo použít čip MCP9700T-E/TT, který byl připojen na poslední volný pin AD převodníku. V rohu plošného spoje zůstalo místo přibližně o velikosti 8 mm. To bylo využito pro LED diody, které se mohou na dronu hodit, například pro identifikaci za temných podmínek.

## 18.10 LDO

Tento plošný spoj představuje pouze výkonovou část a řídící část, která je v této bakalářské práce zastoupena FPGA, potřebuje napájení nezávislé na stavu baterie. Na plošný spoj byl tedy umístěn jeden napěťový regulátor L7805CDT-TR poskytující napětí 5 V a proud 1,5 A.

Všechny referenční napětí přivádím z FPGA a mají hodnotu 3.3 V, což zajišťuje, že digitální výstup AD převodníku (SPI rozhraní) bude pracovat na napěťové úrovni 3,3 V, což je úroveň, na které pracují vstupy FPGA.

Reference pro analogový vstup ADC a pro DRV8305 by mohla být i 5 V, ale sjednocení hodnoty zjednodušuje realizaci. Plošný spoj má dvě diody připojené na

piny pro libovolné použití. Při umisťování komponent je důležité dbát na jejich správnou polohu. Kritické je umístění kondenzátorů co nejblíže příslušnému obvodu. Umístění bylo provedeno skutečně nejblíže, jak jen to bylo možné.

## 18.11 Výběr vhodné technologie rezistorů [40][41][42]

Typ rezistoru určuje, jak moc se rezistor zahřívá při určitém proudu, jak zvládá krátkodobé přetížení a jak silný šum způsobuje. Základní kategorie rezistorů:

- **AntiPulse** – proti přetížení **velkým** a **dlouhodobým** výkonem
- **AntiSurge** – proti přetížení vysokým napětím s krátkou dobou účinností – ESD
- **ThickFilm (tlustovrstvé) -** vyráběny technologií, při které je na keramické tělísko nanesena tlusto-vrstvá odporová vrstva. Výsledný odpor je upravený vybroušením drážky do nanesené vrstvy**.** Tato technologie je levná, proud tekoucí drážkou vytváří **hodně ztrátového tepla.**
- **ThinFilm (tenkovrstvé)** odporová vrstva je naprášena a tvoří ji čistý kov. Výsledného odporu se dosahuje laserovým vybroušením v podobě cest, jež jsou ekvivalentně distribuovány po celé ploše rezistoru.
- **MELF (Metal electrode leadless face) -** dlouhodobá stabilita, odolnost proti vlhku, odolnost vůči účinkům teplotních cyklů**.** Tyto dobré vlastnosti jsou způsobené tím, že tvar rezistoru je cylindrický, a jeho plocha je tedy daleko větší, což umožnuje lepší distribuci tepla.

## 18.12 Gate rezistor [40][41][42]

Technická specifikace doporučuje použít hodnotu **30 Ω.** Byla zvolena nepříliš standartní **velikost 1210 (3225 metricky)**, jelikož rezistory byly umisťovány na PCB jako poslední a dalo se využít co nejvíce zbylého místa. Nejedná se o aplikaci, kde požadujeme přesnou hodnotu a stabilitu s teplotou, jelikož je volba hodnoty přibližná, a byl tedy zvolen tlusto-vrstvý rezistor s ochranou proti ESD, jelikož cenový rozdíl tvoří pouze 30 % oproti standartní verzi, a navíc právě MOSFET tranzistory jsou komponenty citlivé na ESD.

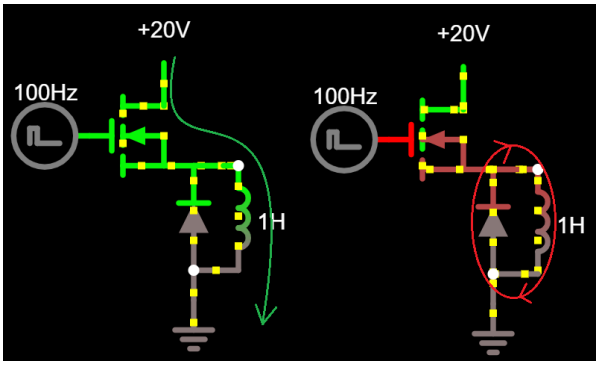
# 19 SOUHRN SOUČÁSTEK

| Komponent | Popis | Hodnota | Velikost | Počet | Cena | Reference |
|---|---|---|---|---|---|---|
| BLDC-DRIVER-PCB | Plošný spoj pro ovládání BLDC | 50 V, 10 A | 29.3x46.6 | 1x | 220 | |
| DRV8305NPHPR | 3fázový budič motoru | | HTQFP-48 | 1x | 110Kč | U1 |
| IPD14N06S280ATMA2 | MOSFET-N tranzistor | 55 V, 17 A | D-PAK | 6x | 103Kč | U2 až U7 |
| V8PAL45-M3/I | Shottkyho dioda | 8 A, 45 V, 570 mV | DO-221BC | 6x | 145Kč | CR2 až CR7 |
| ADC108S022CIMT/NOPB | AD převodník | 8 ch, 10 bit, 200 kSPS | TSSOP | 1x | 120Kč | U8 |
| L7805CDT-TR | Lineární napěťový regulátor | 5 V, 1.5 A | D-PAK | 1x | 21Kč | CR1 |
| EEEFT1H331AP | AL elektrolytický kondenzátor | 330 µF, 50 V | | 2x | 56Kč | C10, C12 |
| CRE2512-FZ-R007E-3 | Rezistor pro měření proudu | 0.007 Ω, 3 W, ±3 % | 2512 | 3x | 66Kč | R7, R12, R15 |
| ERJP14F1000U | Gate rezistor | 100 Ω, 0.5 W, ±1 % | 1210 | 6x | 44Kč | R17 až R22 |
| MCP9700T-E/TT | Teploměr | ±4 °C | SOT-23 | 1x | 10Kč | U9 |
| M40-3200845R | PinHeader - 16 pinů, 2 řady, | 1.0 mm | | 1x | 50Kč | J1 |
| C1608X7R1C105K080AC | Vícevrstvý keram. kondenzátor | 1 µF, 16 V, ±10 % | 0603 | 3x | 10Kč | C8, C9, C11 |
| GRM31CR71H475KA12L | Vícevrstvý keram. kondenzátor | 4.7 µF, 50 V, ±10 % | 1206 | 1x | 26Kč | C3 |
| C0603C473K1RACTU | Vícevrstvý keram. kondenzátor | 47 nF, 100 V, ±10 % | 0603 | 2x | 18Kč | C6, C7 |
| C0603C104J4RACTU | Kondenzátor KEMET | 0.1 µF, 16 V, ±5 % | 0603 | 7x | 21Kč | C9 14 15 18 20 22 24 |
| MCSH18B102K160CT | Kondenzátor MulticomPro | 1 nF, 16 V, ±10 % | 0603 | 3x | 10Kč | C8, C13, C16 |
| GRM188R61H225KE11D | Kondenzátor Murata | 2.2 µF, 50 V, ±10 % | 0603 | 1x | 10Kč | C5 |
| C1608X5R1H105K080AB | Kondenzátor TDK | 1 µF, 50 V, ±10 % | 0603 | 3x | 20Kč | C11, C16, C19 |
| MCWR06X101 JTL | Rezistor MulticomPro | 100 Ω, 0.1 W, ±5 % | 0603 | 1x | 0.2Kč | R3 |

# 20 SIMULACE MĚNIČE [44]

Pro simulaci bylo použito online prostředí falstad.com. Simulace byla provedena za účelem lepšího pochopení funkce obvodu. Bylo zjištěno, že simulaci hodně ovlivňuje nastavená indukčnost induktorů. Při malé hodnotě indukčnosti se neprojeví zpětný proud, naopak při velké zpětný proud nezastaví do dalšího sepnutí.

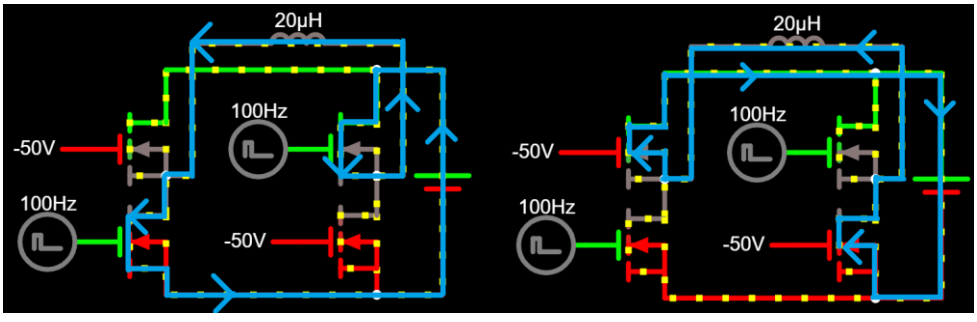
## 20.1 Simulace chování cívky při spínání




**Obrázek 48 Zapojení simulace spínání cívky**

V první simulaci byla ověřena funkčnost tzv. free wheeling diody, jež bývá nazývaná také jako fly back dioda. Je demonstrováno chování cívky při spínání. V levé části Obrázek 48 můžeme vidět sepnutý MOSFET tranzistor, jímž prochází proud do cívky. Dioda paralelně připojená má na katodě kladné napětí vůči anodě, jež je připojená na GND. Dioda je tedy polarizována v záporném směru a nevede žádný proud.

V pravé části obrázku vidíme obvod ve stavu, kdy došlo k uzavření MOSFET tranzistoru. Cívka má v sobě naakumulovanou energii a je schopná vytvořit jakkoliv vysoké napětí na svém výstupu a jakkoliv vysoké záporné napětí na svém vstupu, jen aby odvedla naakumulovanou energii. Cívka tedy vytvoří takové napětí, že se paralelně připojená fly back dioda stane polarizovanou v kladném směru a proud obvodem začne téct, tak jak je vyznačeno na obrázku. Energie z cívky se vyzáří na přechodu diody.

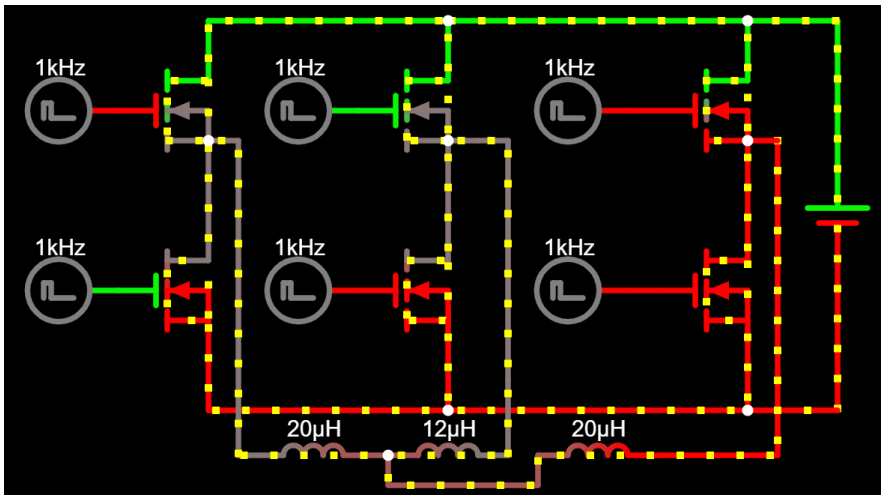



**Obrázek 49 Zapojení simulace měniče pro DC motor**

## 20.2 Simulace můstku s dvěma větvemi

Simulace byla provedena v době, kdy byl předpoklad, že se bude spínat horní i spodní tranzistor. V levé části Obrázek 49 lze vidět otevřený levý spodní a pravý horní tranzistor a proud teče tak, jak je vyznačeno. V pravé části obrázku vidíme uzavřené všechny tranzistory. Tato simulace potvrzuje teoretické předpoklady ze [33].

## 20.3 Simulace 3fázového měniče




**Obrázek 50 Zapojení simulace 3fázového měniče**

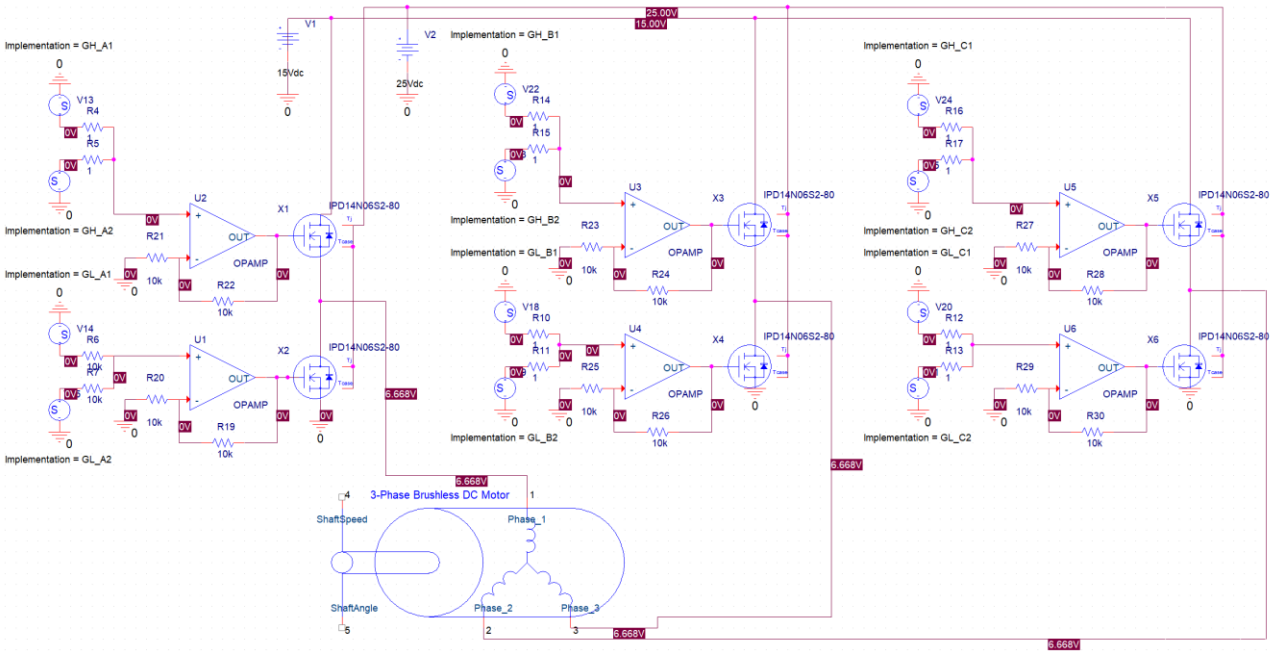
Po ověření chování menších a zjednodušených částí obvodu bylo přistoupeno k simulaci celého 3fázového měniče. Bylo zjištěno přibližné chování obvodu, ale jedná se spíše skutečno simulaci obecného chování než konkrétní model. Došlo se k následnému závěru:

Cívka vytvoří v místě připojení motoru (mezi horním a spodním tranzistorem větve měniče) záporné napětí takové, aby proud prošel parazitní diodou.

Vypneme-li horní tranzistor, cívka připojená na jeho source vytvoří takové záporné napětí, že zem, na kterou je připojena parazitní dioda spodního tranzistoru, bude kladnější a proud začne protékat ze země do cívky – krátkodobě.

Dále byl proveden průzkum vhodného simulátoru s následujícími požadavky. Možnost použití SPICE modelů tranzistorů, jež jsou použity v návrhu PCB. Možnost smíšené digitálně analogové simulace. Simulátor, který již obsahuje model BLDC motoru.

# 21 SIMULACE V ORCAD CAPTURE [35]

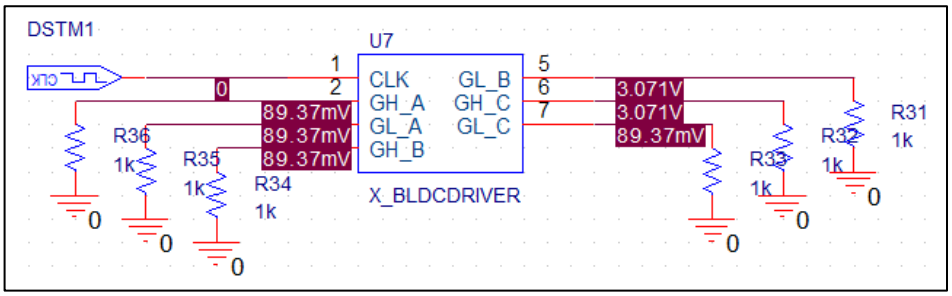



**Obrázek 51 Zapojení měniče v OrCadCapture**

Nejvhodnějším simulátorem se zdál být OrCad Capture od Cadence. Byla získána zkušební trial verze. Podařilo se importovat do prostředí model tranzistoru, který je použit na desce plošného spoje. Analogová část šla sestavit snadno, jelikož prostředí nabízí obecné SPICE modely obecných pasivních i aktivních komponent. Model BLDC motoru bylo potřeba vyhledat v konkrétní knihovně. Složitější částí, která nebyla dokončena kvůli vysoké úrovni náročnosti a také kvůli potřebě pracovat na PCB, na jejíž dodání se čekalo, bylo vytváření tzv. stimulu, což je v našem případě digitální část obvodu. Jednoduchý stimulus lze vytvořit použitím pulzujících zdrojů napětí, spojených pomocí operačního zesilovače v zapojeného pro sčítání (Obrázek 51) tímto způsobem však nelze nasimulovat chování programu napsaného ve VHDL pro FPGA. Pro vytvoření pokročilejšího stimulu byl použit analogový blok VSTIM z knihovny ANALOG. Bylo zjištěno, že tato možnost také není dostačující a mohla být přirovnána k signálovému generátoru – tedy naprogramovaný průběh se periodicky opakuje bez možnosti změny frekvence či amplitudy.

Informace o digitální simulaci a návody na její provedení jsou velmi těžko dostupné. Došlo se k možnému způsobu řešení, ale vzhledem k časové tísni nebylo dosaženo požadované úrovně stimulu. Tato možnost umožňuje vytvořit stimulus v jazyce C++. Vytváření probíhá následovně: Jedním z programů instalovaných jako součást balíčku OrCad je tzv. PSpice Model Editor, který umožňuje vygenerovat za použití tzv. *DMI Template Code Generator* dynamickou knihovnu, k ní projekt pro

Visual Studio Community a do příslušných zdrojových souborů dopsat kód ovládající výstupy. Je možné vytvářet jak kombinační, tak sekvenční logiku. Taktéž je možné vytvoření globálních proměnných, kterými lze z prostředí OrCad Capture předávat parametry do programu. Pro použití ve SPICE simulaci je z knihovny



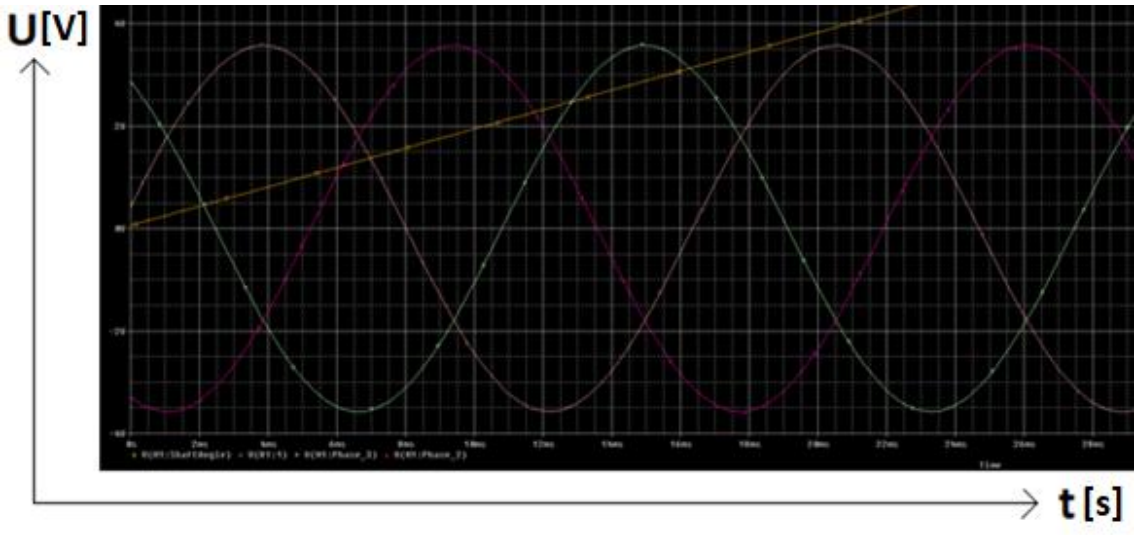

**Obrázek 52 Simulace C++ kódu**

potřeba vygenerovat schématickou značku (Obrázek 52) na kterou se napojí měnič. V případě sekvenční logiky je potřeba připojit i hodinový signál.

Vytvoření stimulu tedy vyžaduje přepsání celého řídícího programu z jazyka VHDL do jazyka C++, a to je velmi náročná záležitost, přihlédneme-li ke komplexnosti aktuálního řídícího programu. Program ve VHDL se také v průběhu času ladění mění a bylo by potřeba změny zrcadlit i do kódu v C++ pro simulaci.

Simulace byla dotažena do bodu, kdy se podařilo zprovoznit přenášení hodnot nastavovaných v kódu na hodnoty čtené na sondách při časové simulaci obvodu. Fungovaly ovšem pouze některé. Závěrem je třeba konstatovat, že se jedná o velice komplexní simulační systém vyžadující dlouhodobou praxi.

Bylo také zjištěno, že model BLDC motoru, poskytovaný jako součást programu, nemá trapézový průběh zpětného elektromotorického napětí, nýbrž průběh harmonický. Toto ověření lze snadno provést připojením napětí na mechanický vstup motoru a následné měření napětí přes odpor na fázích ve SPICE simulaci. Obrázek 53 ukazuje výsledné napětí na fázích při točení mech. vstupem.




**Obrázek 53 Zpětné elektromotorické napětí modelu BLDC motoru v OrCad**

# 22 OSAZENÍ PLOŠNÉHO SPOJE [34]

Deska je velmi malá a osazení je možné provést ručně. Je třeba dbát na určitá pravidla a zvážit způsob osazení. Obecně máme v domácím prostředí dvě možnosti:

## 22.1 Osazení mikropájkou

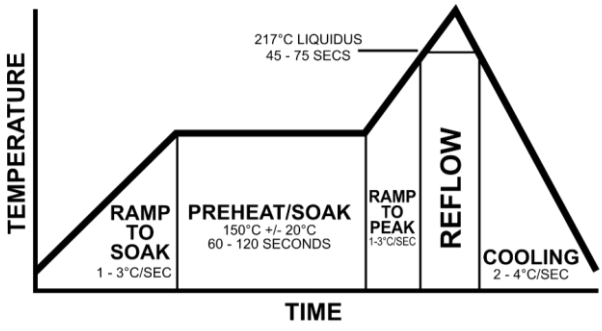
Jedná se o časově náročnější variantu oproti osazení horkovzduchem, jelikož osazení probíhá po jedné součástce. Vzhledem k delší době osazování je plošný spoj vystavený delší dobu tepelné námaze, která může poškodit například lepidla spojující měděnou vrstvu se středem plošného spoje. Tato metoda vyžaduje velmi bohatou praxi, protože hrozí přehřátí součástek při nedodržení maximální doby pájení.

## 22.2 Osazení horkovzduchem

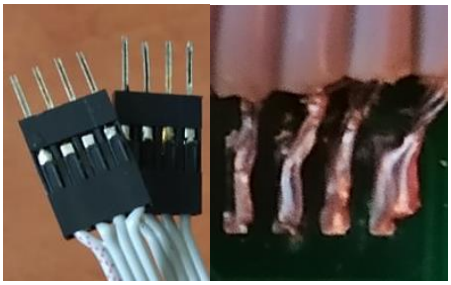
Velkou výhodou je fakt, že u této metody nedochází k fyzickému doteku se součástkami. Samotný proces pájení je velmi rychlý, jelikož každou komponentu stačí nahřát jen pár sekund. Delší je přípravná fáze, kdy je potřeba na každou plošku nanést pájecí pastu, což je směs, jež se po zahřátí rozpustí na pájku a tavidlo. Po nanesení pasty je taktéž potřeba pinzetou umístit součástky co nejpřesněji na své místo. Nevýhodou je, že pájení probíhá vzduchem, který při neopatrnosti a špatném nastavení rychlosti může součástky z desky odfouknout.

## 22.3 Teplota pájení [34]

Ať už jedná o první, nebo druhý způsob osazení, je potřeba se řídit teplotní křivkou (viz obr. 22.3-1). Postupné předehřátí, ochlazení a dodržování časových intervalů v těchto fázích je důležité, protože materiály, z nichž se skládá plošný spoj, mají různé teplotní roztažnosti a při skokové změně teploty by došlo k poškození PCB, např. u propojení širokých 0.2 mm a u jejich napojení na VIAs.




**Obrázek 54 Teplotní křivka pro pájení [34]**



**Obrázek 55 Zapojení plochého kabelu**

## 22.4 Čištění plošného spoje

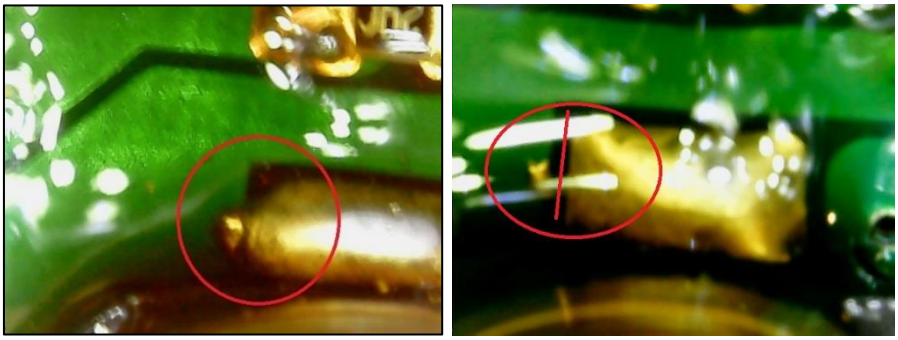
Při nanášení pasty bez šablony může po zahřátí vzniknout více pájky, než se čekalo a je potřeba spoj opravit mikropájkou. Použití mikropájky je nevyhnutelné také při úpravách hodnot součástek, nebo případné výměně nefunkčních obvodů. K tomuto způsobu pájení je potřeba tavidlo (např. kalafuna). Vedlejším efektem je, že plošný spoj zůstává lepivý a zamezuje inspekci správnosti osazení pomocí mikroskopu nebo kamery. K rozpuštění kalafuny je potřeba rozpouštědlo, které nesmí být příliš silné, aby nepoškodilo nepájecí masku plošného spoje. Ideální varianta, používaná v technické praxi, je použití isopropyl alkoholu, který není agresivní vůči plošnému spoji. Nakonec byl použit líh, protože je snadno dostupný.

## 22.5 Připojení plochého kabelu

Plochý kabel byl připájen přímo na piny (Obrázek 55), jelikož driver bude umístěn na křídle dronu hned pod vrtulí a při použití pinů by se nevešel na výšku.

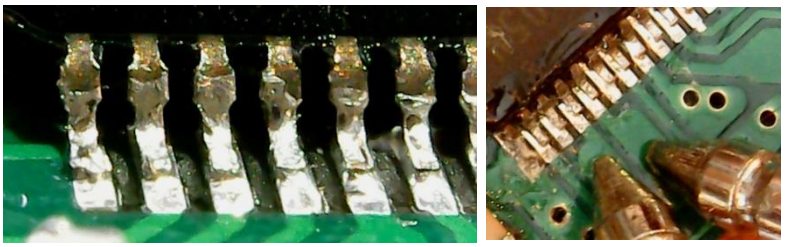
## 22.6 Kontrola správnosti zapájení součástek na plošný spoj

Pro kontrolu, zda jsou součástky správně zapájené, bylo použito jak vizuální kontroly, tak kontroly elektrickým prověřením propojení pomocí multimetru. Při prověřování multimetrem byly napájecí přívody indikovány jako zkrat. Ukázalo se, že zkrat byl způsobený poškozenou nepájivou maskou (Obrázek 57).



**Obrázek 57 Poškozená pájecí maska způsobující zkrat**

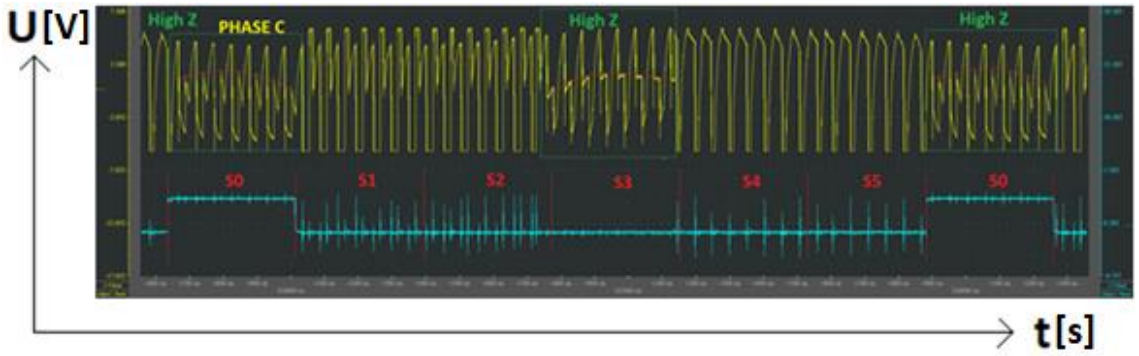
Při pájení driver DRV8305 bylo potřeba řešit přebytek cínu, který spojoval vývody obvodu k sobě. Odstranění bylo provedenou za použití pájecího knotu, což je knot z mědi určený k odsávání přebytečného cínu.



**Obrázek 56 Přiblížení zapájení DRV8305**

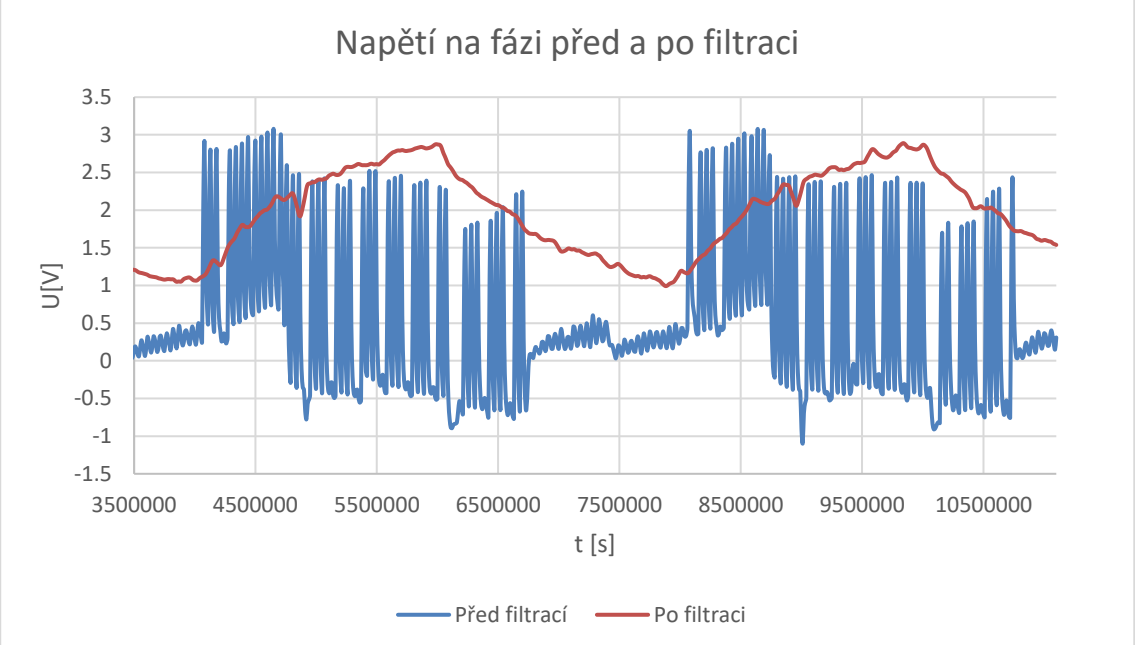
# 23 MĚŘENÍ NA VÝSLEDNÉM ZAPOJENÍ

Průběh napětí na fázi při spínání horních i spodních tranzistorů – metoda ovládání můstku, při které dochází ke dvojnásobným ztrátám spínáním a znemožňuje měřit zpětné elektromotorické napětí. Modrý (spodní) signál je aktivní v momentě, kdy je fáze připojená na stav vysoké impedance. Lze jednoznačně vidět, že zpětné elektromotorické napětí nelze z horního signálu vyčíst.



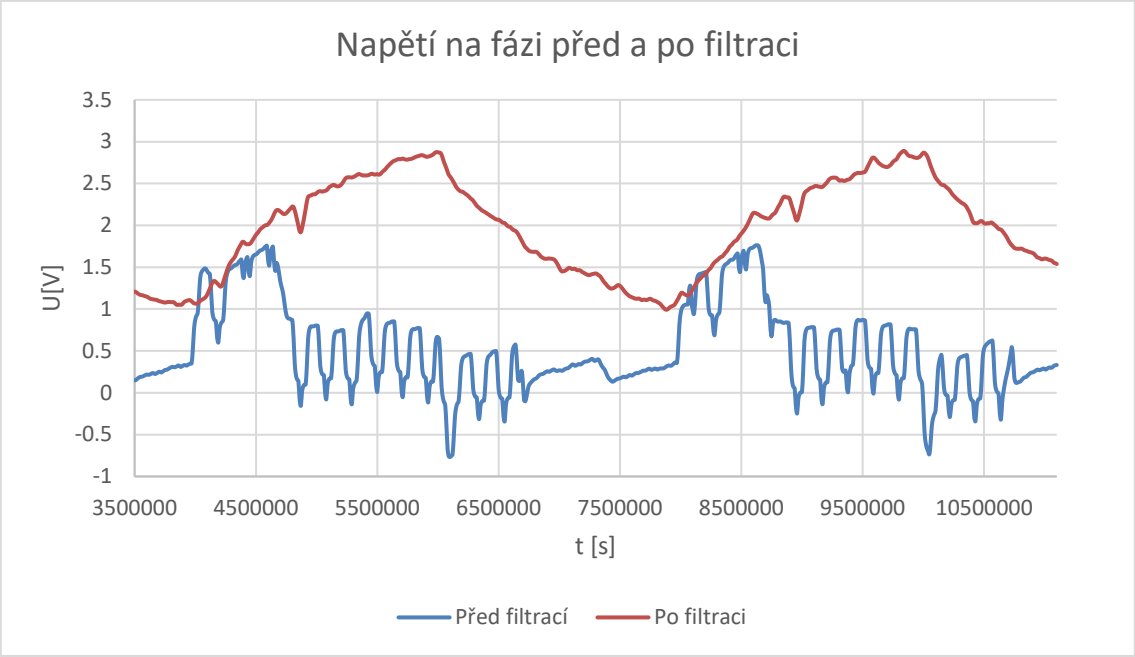

**Obrázek 58 Průběh napětí na fázi při použití PWM pro horní i spodní tranzistory**

Následující oObrázek 58 ukazuje signál na fázi měřený přímo a na vstupu do AD převodníku za dolnofrekvenční propustí.

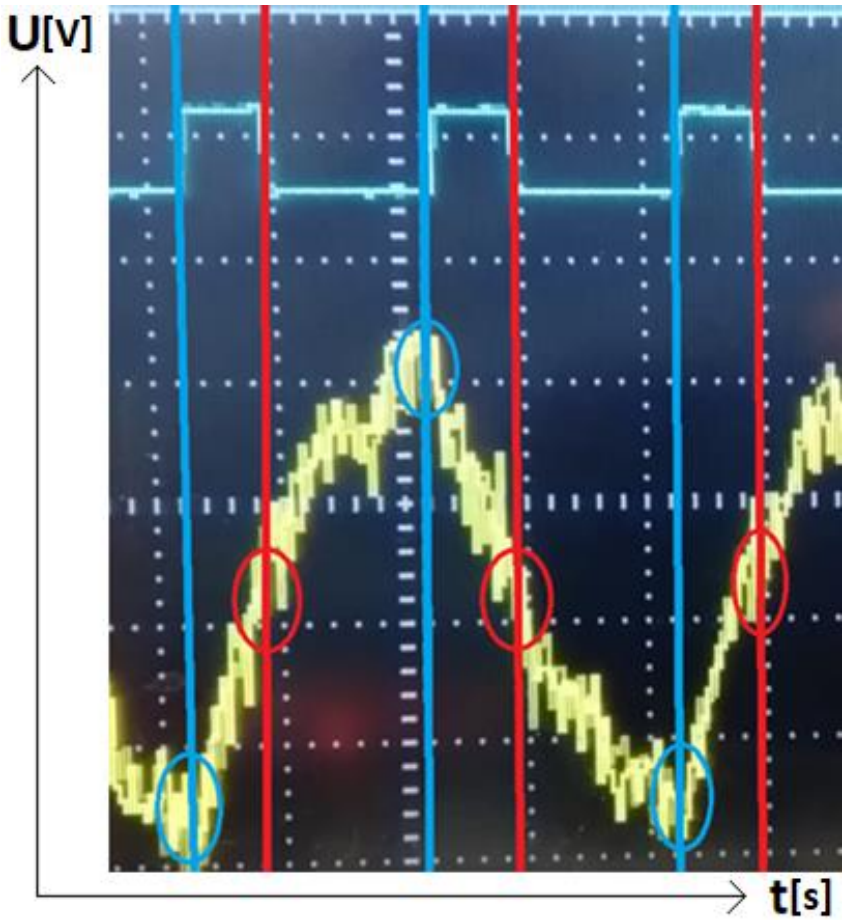



**Obrázek 59 Průběh napětí na fázi po aplikaci PWM na horní tranzistory měniče**

Následující graf ukazuje stejnou závislost, ale signál před filtrací je numericky zprůměrován vždy z 10 posledních vzorků. Měřením bylo zjištěno, že změnou rychlosti v důsledku potřeby větší střídy dochází k posunu signálu po filtraci na vstupu AD převodníku směrem k vyšším napětím.

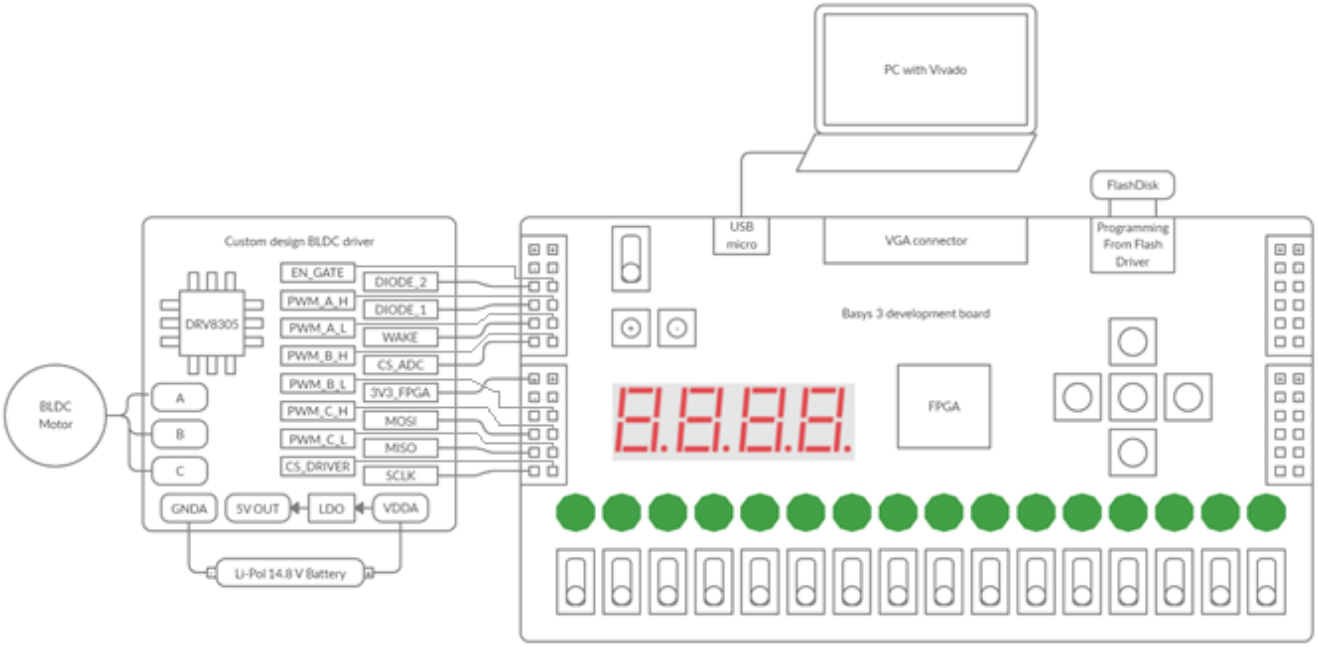



**Obrázek 61 Signál z obrázku 62 po filtraci**

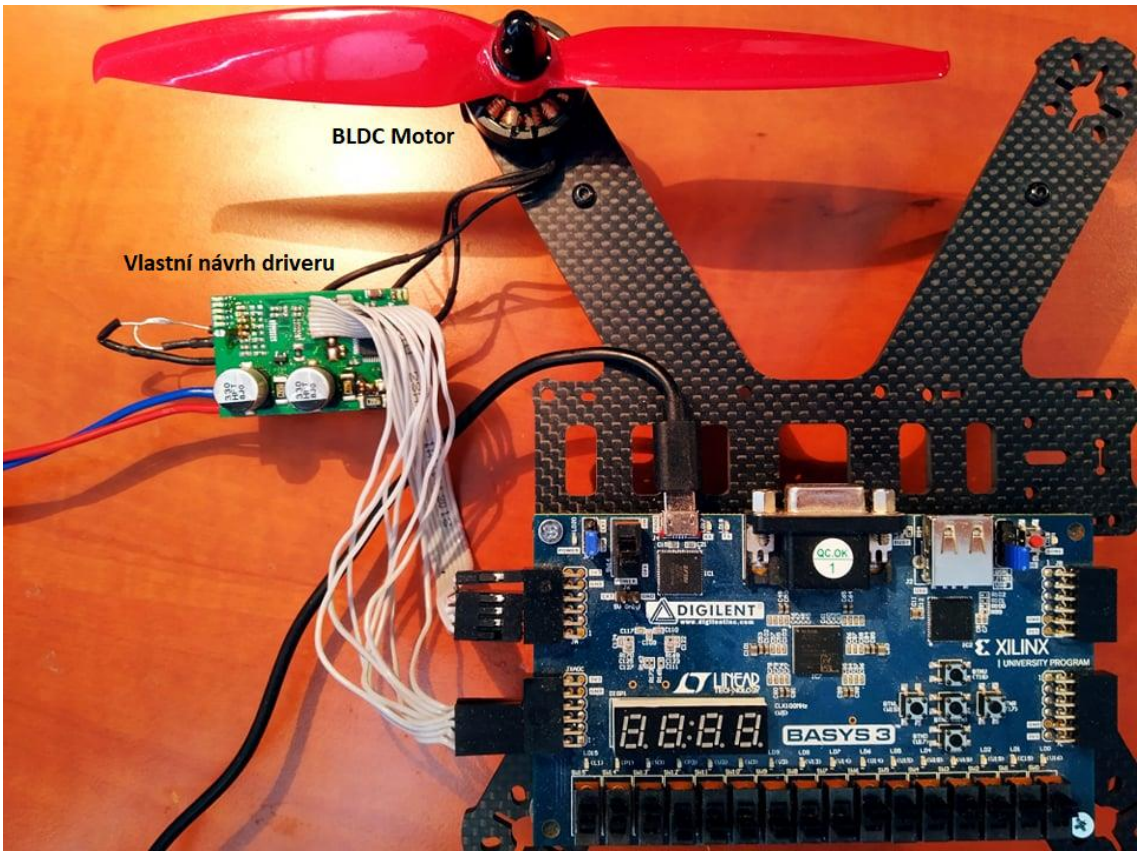



**Obrázek 60 Průběh napětí fáze za filtrem na vstupu ADC**

# 24 KOMPLETNÍ BLOKOVÉ SCHÉMA




**Obrázek 63 Výsledná realizace blokově**




**Obrázek 62 Výsledná realizace fotografie**

# ZÁVĚR

V semestrální práci byl vytvořen model pro řízení za použití vektorového řízení, což je výpočetně nejnáročnější forma řízení motoru. Pro fyzickou realizaci v bakalářské práci bylo vybráno řízení pomocí šestikrokové komutace, která se zdařila ve formě ovládání – tedy řízení v otevřené smyčce.

Byly realizovány dva různé drivery. Driver používající obvod IR2110 byl sestaven na univerzálním pájecím poli a posloužil k odladění spouštěcí sekvence pro motor. Druhý driver, používající technologicky pokročilý obvod DRV8305, byl realizován jako plošný spoj integrující AD převodník pro snímání napětí na fázích a proudů větvemi měniče. Driver je tedy možné využít i pro vektorové řízení, které vyžaduje znalost proudů.

Z možných druhů spínání měniče byl zvolen ten, který je vhodný pro čtení zpětného elektromotorického napětí. Akvizice dat z AD převodníku vyžaduje implementaci filtrů, protože signál je i přes analogový filtr, představující dolní propust, plný rušivých složek.

Motor dosahuje v řízení v otevřené smyčce přibližně patnácti tisíc otáček za minutu – testování probíhá s vrtulí. Driver se příliš nezahřívá a můžeme tedy říct, že bylo ověřeno správné dimenzování součástek. Odebíraný proud nepřekračuje při konstantní rychlosti 1.5 A. Při změnách rychlosti je potřeba dodat více proudu, a to až 5 A.

Byla implementována komunikace přes SPI, a to včetně systému chybových kódů, které jsou generovány na základě přijatých dat z DRV8305. Hardwarový popis pro jeden driver spotřebuje 1542 LUT, 1540 FF. Samotné připojení driveru potřebuje 16 IO pinů.

Řízení motorů je velmi komplexní problematika a tato bakalářská práce spolu se semestrální prací realizovala jeden způsob řízení simulací a druhý fyzickou implementací.

Návrh driveru se na testovacím prototypu ukázal jako funkční. Mohu tedy osadit všechny zbývající desky a použít driver na vlastním projektu, který se zabývá kompletním sestavením dronu.

# Literatura


[1] MAKÓWKA, David. FPGA modul pro řízení BLDC motorů [online]. Brno, 2020 [cit. 2020-06-07]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/123165. Semestrální práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Soběslav Valach.

[3] Technická specifikace pro DRV8305
http://www.ti.com/lit/ds/symlink/drv8305.pdf
[online] [Cit. 5.1.2020]

[4] Seminář od Texas Instruments o možných druzích řízení motorů
https://training.ti.com/teaching-old-motors-new-tricks-c2000
[online] [Cit. 5.1.2020]

[5] Popis topologie SVM
https://www.youtube.com/watch?v=5x73LjZQ21o
[online] [Cit. 5.1.2020]

[8] Tutoriál možné realizace SVM PWM v MATLAB Simulink
https://www.youtube.com/watch?v=QMSWUMEAejg
[online] [Cit. 5.1.2020]

[6] Technická dokumentace Simulink komponenty **ABC to DQ**
https://www.mathworks.com/help/physmod/sps/powersys/ref/abctodq0dq0toabc.html
[online] [Cit. 5.1.2020]

[7] Technická dokumentace Simulink komponenty **DQ to AlphaBeta**
https://www.mathworks.com/help/physmod/sps/powersys/ref/alphabetazerotodq0dq0toalphabetazero.html
[online] [Cit. 5.1.2020]

[9] Princip funkčnosti analogově-digitálního převodníku typu **SigmaDelta**
https://www.analog.com/en/technical-articles/sigma-delta-conversion-used-for-motor-control.html#
[online] [Cit. 5.1.2020]

[11] Informace o pamětech v FPGA
https://www.xilinx.com/support/documentation/application_notes/xapp464.pdf
[online] [Cit. 5.1.2020]

[12] Fischer P. High Performance Brushless DC Motor Control. School of Engineering &Technology CQUniversity Australia. May 2014
http://www.ti.com/lit/an/sprt703/sprt703.pdf
[online] [Cit. 5.1.2020]

[13] Obrázek SVM modulace
https://en.wikipedia.org/wiki/Space_vector_modulation#/media/File:Space_Vector_Modulation.gif
[online] [Cit. 5.1.2020]

[14] Vektorové řízení za použití Simulinku
https://www.youtube.com/watch?v=9D79OlyyZAU
[online] [Cit. 5.1.2020]

[15] Simulace Parkovy a Clarkovy transformace v Simulinku
https://www.youtube.com/watch?v=Y2gUb0zdS7M
[online] [Cit. 5.1.2020]

[16] Přehled FPGA čipů společnosti Xilinx
https://www.xilinx.com/support/documentation/selection-guides/7-series-product-selection-guide.pdf
[online] [Cit. 5.1.2020]

[17] Návod na tvorbu C# aplikace pro komunikaci s využitím COM portu
https://www.youtube.com/watch?v=I6uhMIFTF24&fbclid=IwAR0S1nP9lLuetigdUPtQmQxjmR1rqhooh5UWUTHHdHDrMfCnxNUvlE52Z8A
[online] [Cit. 5.1.2020]

[18] Informace o sledovači zpětného elektromotorického napětí
https://e2e.ti.com/blogs_/b/industrial_strength/archive/2014/03/14/have-you-considered-using-back-emf-observers-for-sensor-less-speed-estimation
[online] [Cit. 5.1.2020]

[19] Popis transformací pro vektorové řízení
https://www.youtube.com/watch?v=vdeVVTltr1M
[online] [Cit. 5.1.2020]

[20] Praktické informace pro ladění regulátorů
https://www.youtube.com/watch?v=UuealbABA_Y
[online] [Cit. 5.1.2020]

[21] Kaskádní regulace
https://www.motioncontroltips.com/why-is-the-bandwidth-of-a-servo-control-loop-important/
[online] [Cit. 5.1.2020]

[22] Kaskádní regulace
https://www.motioncontroltips.com/faq-servo-motor-current-velocity-position-loops-bandwidths/
[online] [Cit. 5.1.2020]

[23] Rozdíl mezi BLDC a PMSM
https://www.youtube.com/watch?v=R7E61d9cLi0
[online] [Cit. 5.1.2020]

[24] Jak implementovat sériovou komunikace na Basys 3
https://www.youtube.com/watch?v=Fms2Qwkbu1g
[online] [Cit. 5.1.2020]

[25] Požadavky na AD převodníky pro řízení motorů
https://www.electronicdesign.com/technologies/dsps/article/21752174/adcs-lend-flexibility-to-vector-motorcontrol-applications
[online] [Cit. 5.1.2020]

[26] Rozdíl mezi distribuovanou a blokovou RAM
https://www.youtube.com/watch?v=T4khi8MmoVc
[online] [Cit. 5.1.2020]

[27] Popis funkčnosti LDO regulátoru
https://www.youtube.com/watch?v=cM7t1Mpu7s4
[online] [Cit. 5.1.2020]

[28] Zdroj pro symboly a footprinty součástek
https://app.ultralibrarian.com/search?queryText=DRV8305&page=1
[online] [Cit. 5.1.2020]

[29] Návod jak tvořit filtry v Matlabu
https://www.youtube.com/watch?v=VFt3UVw7VrE
[online] [Cit. 5.1.2020]

[30] SVM v Simulinku
https://www.youtube.com/watch?v=39HX4qB3M74
[online] [Cit. 5.1.2020]

[31] Detekce zero crossing
https://www.st.com/resource/en/application_note/cd00043112-back-emf-detection-during-pwm-on-time-by-st7mc-stmicroelectronics.pdf
[online] [Cit. 30.5.2020]

[33] Druhy PWM techniky pro komutaci motorů
https://e2e.ti.com/blogs_/b/industrial_strength/archive/2012/03/26/so-which-pwm-technique-is-best-part-2
[online] [Cit. 30.5.2020]

[34] Teplotní profil pro pájení přetavením
https://en.wikipedia.org/wiki/Reflow_soldering#/media/File:RSS_Components_of_a_Profile1.svg
[online] [Cit. 30.5.2020]

[35] Příručka pro digitální a analogovou simulaci v OrCad Capture
https://www.pspice.com/resources/application-notes/device-model-interface
[online] [Cit. 30.5.2020]

[36] Význam vrstev v KiCad editoru pro plošný spoj
https://forum.kicad.info/t/what-is-the-meaning-of-the-layers-in-pcb-new-and-in-the-footprint-editor/9688
[online] [Cit. 30.5.2020]

[37] Význam použití parazitní diody MOSFET tranzistorů
https://toshiba.semicon-storage.com/ap-en/semiconductor/knowledge/faq/mosfet/is-it-ok-to-use-the-body-diode-parasitic-diode-between-the-drain.html
[online] [Cit. 30.5.2020]

[38] Výpočet pásma filtru napětí na fázi
http://ww1.microchip.com/downloads/en/AppNotes/doc8012.pdf
[online] [Cit. 30.5.2020]

[39] Faradayův zákon
https://en.wikipedia.org/wiki/Faraday%27s_law_of_induction
[online] [Cit. 30.5.2020]

[40] Přehled typů rezistorů
https://www.koaglobal.com/product/howtouse/resistor/r_feature?sc_lang=en
[online] [Cit. 30.5.2020]

[41] Test tepelné odolnosti různých typů rezistorů
https://www.youtube.com/watch?v=cdNnEbfZ9us
[online] [Cit. 30.5.2020]

[42] Informace k rezistoru typu MELF
https://www.electronics-notes.com/articles/electronic_components/resistors/melf-smd-resistor.php
[online] [Cit. 30.5.2020]

[43] Evaluační modul pro DRV8305
https://www.ti.com/tool/BOOSTXL-DRV8305EVM
[online] [Cit. 30.5.2020]

[44] Online simulační prostředí Falstad
https://www.falstad.com/circuit/circuitjs.html
[online] [Cit. 30.5.2020]

[45] Technická specifikace pro ADC108S102
http://www.ti.com/lit/ds/snas336b/snas336b.pdf?ts=1591528723269&ref_url=https://www.google.com/
[online] [Cit. 30.5.2020]

# Ostatní zdroje

[101] VHDL Entita pro SPI rozhraní
https://github.com/nandland/spi-master
(Cit. 5.1.2020)

[102] Binary to BCD converter
https://allaboutfpga.com/vhdl-code-for-binary-to-bcd-converter/
(Cit. 5.1.2020)

[103] SPI Master
https://www.digikey.com/eewiki/pages/viewpage.action?pageId=4096096
(Cit. 8.6.2020)

[104] SPI Slave
https://www.digikey.com/eewiki/pages/viewpage.action?pageId=7569477
(Cit. 8.6.2020)

[105] I2C
https://www.digikey.com/eewiki/pages/viewpage.action?pageId=1012534
(Cit. 8.6.2020)

# Seznam symbolů, veličin a zkratek

| | | |
|---|---|---|
| FEKT | - | Fakulta elektrotechniky a komunikačních technologií |
| VUT | - | Vysoké učení technické v Brně |
| BLDC | - | Brushless DC Motor |
| EC | - | Electronically commutated |
| FPU | - | Floating point unit |
| LUT | - | Lookup Table – Generátor logické funkce |
| BRAM | - | Block RAM – pamět s náhodným přístupem uspořádaná do bloků |
| RAM | - | Random Access Memory |
| SNR | - | Signal to noise ratio |
| kSPS | - | Kilo samples per second |
| MOSI | - | Master out slave in |
| MISO | - | Master in slave out |
| MSB | - | Bit s nejvyšší váhou |
| LSB | - | Bit s nejnižší váhou |
| INL | - | Integrální nelinearita |
| DNL | - | Diferenciální nelinearita |
| PGA | - | Programmable gain amplifier – programově nastavitelný zesilovač |
| LUT | - | FPGA Look Up Table |
| CLB | - | Configurable logic block – nastavitelný logický blok |
| SR | - | Shift register – posuvný registr |
| FIFO | - | Posuvný registr – vysouvá hodnoty ve stejném pořadí jako při vstupu |
| BP | - | Bakalářská práce |
| HW | - | Hardware |
| HDL | - | Hardware description language |
| VDDA | - | V této práci označení pro nap. napětí analogových obvodů měniče |
| MOSFET | - | Metal Oxide Semiconductor Field Effect Transistor |
| IC | - | Integrated circuit – integrovaný obvod |
| EDA | - | Electronic design automation |
| LDO | - | Linear-dropout regulator |
| SPICE | - | Simulation Program with Integrated Circuit Emphasis |

# Seznam příloh